İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ITU-GRAM+ : WEB TABANLI GRID ARA KATMAN YAZILIMI

YÜKSEK LİSANS TEZİ

G.Bengi ŞENDUR

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

HAZİRAN 2012

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ITU-GRAM+: WEB TABANLI GRID ARA KATMAN YAZILIMI

YÜKSEK LİSANS TEZİ

G.Bengi ŞENDUR
(504081515)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Yard Doç. Dr. D. Turgay ALTILAR

Haziran 2012

İTÜ, Fen Bilimleri Enstitüsü'nün 504081515 numaralı Yüksek Lisans Öğrencisi **G.Bengi ŞENDUR** ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"ITU-GRAM+:Web Tabanlı Grid Ara Katman Yazılımı "** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Yard Doç Dr. D. Turgay ALTILAR** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Nadia ERDOĞAN** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Can Özturan Boğaziçi Üniversitesi	.....

Teslim Tarihi : **Mayıs 2012**
Savunma Tarihi : **Haziran 2012**

Eşime,

ÖNSÖZ

Tez çalışmam süresince desteğini hiç eksik etmeyen, anlayışlı tavrıyla zevkli bir çalışma ortamı sağlayan değerli hocam Sayın Yard. Doç Dr. D.Turgay Altılar'a teşekkürlerimi sunarım.
Aileme, hayatımın her alanında olduğu gibi eğitimim süresinde gösterdikleri her türlü destek için ayrıca minnettar olduğumu belirtmeliyim.
Özellikle eşim Burhan Türkel'e bu çalışmam sürecinde de gösterdiği sınırsız sabır ve destek için ayrıca şükranlarımı sunarım.

Mayıs 2012 G.Bengi ŞENDUR TÜRKEL

İÇİNDEKİLER

Sayfa

KISALTMALAR

CPU : Merkezi İşlen Birimi
RAM : Rastgele Erişimli Hafıza
FTP : Dosya Transfer Protokolü
GIS : Grid Bilgi Servisi
JVM : Java Sanal Makinesi
ORM : Nesne İlişki Eşleşmesi
SQL : Yapılandırılmış Sorgu Dili
MVC : Model –Görünüm-Denetleyici
OGSA : Açık Grid Servis Mimarisi
WSRF : Web Servis Kaynak Çatısı
VTYS : Veri Tabanı Yönetim Sistemi
POJO : Düz Eski Java Nesnesi
XML : Genişletilebilir İşaretleme Dili
JSP : Java Sunucu Sayfaları
ODBC : Oracle Veritabanı Bağlantısı
CSS : Basamaklı Stil Şablonları

ŞEKİL LİSTESİ

Sayfa

ITU-GRAM+:Web Tabanlı Grid Ara Katman Yazılımı

ÖZET

Bilim ve mühendislik alanındaki birçok uygulama gün geçtikçe karmaşıklaşan hesaplama ve veri ihtiyaçlarını karşılamak için tek bir kaynak üzerinde çalışmaktansa dağıtık kaynakları kullanmaya yönelmiştir. Bu dağıtık kaynakların farklı yönetim sistemleri altında olması, heterojen ve dinamik yapıları üzerinde planlama işleminin yapılmasını zorlaştırmıştır. Bu çalışma içerisinde bu kaynakların en verimli şekilde kullanılabilmesi için standart arayüz ve protokoller ile birlikte yeni teknolojiler kullanılarak bir grid yönetim sistemi tasarlanıp, gerçeklenmesi yapılmıştır. Uygulamanın tümü açık kaynak kodlu yazılımlar kullanılarak geliştirilmiştir. Yazılım geliştirenler açısından bakıldığında kurulumu kolay ve üzerinde geliştirme yapılabilir bir altyapı sağlar.

Kullanıcıların iş talepleri web arayüzünden standart arayüzler ve tanımlamalar ile alınarak, bu ihtiyaçları asgari ölçüde karşılayabilecek ve belirtilen zaman aralığı içerisinde başka herhangi bir işe atanmamış olan kaynakların bir listesi oluşturulur. Bu liste kullanıcının vermiş olduğu bütçe filtresinden geçirildikten sonra kullanıcıya listelenir. Kullanıcının bütçesine en uygun olduğunu düşündüğü kaynakları seçmesi istenir.Bu seçim yapıldıktan sonra seçilen kaynaklar kullanıcının vermiş olduğu iş ile eşleştirilir ve kaynak atanması yapılır. Bu atamalar yapıldıktan sonra çalıştırma zamanı gelen işler web servis teknolojisi kullanılarak uzaktaki kaynak ile iletişme geçilir ve işin çalıştırılması için gereken tüm bilgiler kaynağa parametre olarak verilir. Kaynak sistem, iş ile ilgili gerekli giriş veya çalıştırılabilir program parçalarını içeren dosyaları ftp protokolunu kullanarak kendi yerel sisteme kayıt eder. Kaynak sistem bu dosyaları kullanarak işin çalıştırılmasına başlar ve yaptığı her adım sonrasında işin durumu ile ilgili olarak yine web servis teknolojisini kullanarak, sunucu sisteme bilgilendirmeler gönderir. İş başarıyla tamamlandıktan sonra oluşan çıktı dosyalarını daha önceden belirtilen sisteme ftp aracılığıyla iletir. Son kullanıcılar göndermiş olduğu işlerin son durumlarını sistem üzerinden izleme olanağına sahiptir. Kaynaklar üzerinde doğru seçimler yapılabilmesi için sistem üzerindeki kaynak bilgilerinin güncel olması gereklidir. Bu doğru bilgileri elde etmek amacı ile Grid Bilgi Sistemi bileşeni proje kapsamına eklenmiştir. Yine web servis teknolojisi kullanarak kaynaklar üzerinde servisler tanımlanmış ve bu servisler sunucu sistemlere açılarak kaynak bilgilerin sistem üzerinde güncellenmesi sağlanmıştır.

Bu proje ile kullanıcılara bütçelerine uygun farklı seçenekler sunulmuş, standart arayüzler ve protokoller sağlayarak işin ayrıntıları ile uğraşmaları engellenmiş ve dosya gönderimlerini ağ üzerinde geçen zamanını azaltmak adına seçilecek kaynakların birbirlerine olan uzaklıkları da göz önüne alınmıştır. ITU-GRAM+ ismi de bu nedenle konulmuştur.Kaynak yönetimi dışından birçok artı değer katacak özelliğe sahiptir.Güvenlik ve hata düzeltme ile ilgili bileşenler çalışma kapsamı dışında bırakılmış fakat sağladığı altyapı sayesinde de geliştirilmeye açıktır.

ITU-GRAM+:Web Based Grid Middleware

SUMMARY

Most of science and engineering applications tend to work on distributed resources instead of single resource in order to meet computing and data requirements that are becoming complex day by day. These distributed resources that are under different administrative controls systems and because of their heterogeneous and dynamic structures made the planning process difficult. In this study, a new grid resource manager system (ITU-GRAM+) which is implemented by using only open source software based on trend web technologies. The ITU-GRAM+ allows users to access computational and data resources, submit their jobs and track job status via standard interfaces. Main goal of ITU-GRAM+ is mapping the job resource requests to the resources in a way that will satisfy both the application users and resource owners. In order to satisfy this requirement ITU-GRAM+ is designed to have Resource Discovery, Resource Allocation, Job Submission, Job Execution and Job Monitoring components. Some of these components in ITU-GRAM+ work with SOA architecture to facilitate a communication between resources and resource manager which also gain flexibility to the system in terms of adding new resources and new users. For this communication ITU-GRAM+ exposes different web services to run both on resources and resource manager. In this work, all details about components are described with their additional functionality comparing to existing middleware.

Getting job specific information by users and allow them to reserve and allocate the resources according to these requirements is main goal of this web application. But resources in grid system are not only belongs to system, they have to maintain their daily work issues. For this reason, to keep update information about resources in such system, Grid Information System is implemented.The aim of this component to retrieve resource information such as *available cpu*, *available storage*, *available memory* which could be changed dynamically. The challenging question is that how the system informs the resources about releasing job after reservation of resource is completed. The Job Execution Component is developed to send job information to resources via web services which is exposed by resource.

Currently, the Globus Toolkit is the de facto standard for grid computing because of its wide acceptance and deployment worldwide, even though several alternatives do exist, like Legion and Condor. Limitations of the existing Grid middleware which does not take into account the needs of everyday scientific and business users. Day-to-day computer users and small to medium sized organizations often do not use clusters, and thus for them setting up Grids using the existing middleware is complex Furthermore, Grid enabling their applications is nearly impossible, as they are not easily supported, and this poses a massive barrier to the pervasive adoption of Grid computing by these communities. Grid Middleware is also complex to setup and necessitates a steep learning curve. But ITU-GRAM+ has basic interface to use and it is not necessary to adapt your application for grid system. You can submit your jobs as it is. From the perspective of developers , it is easy to create their own middleware

which provides basic infrastructure to add new functionalities and improve the system themselves. From the perspective of day-to-day computer users, scientists and business users, it is easy to start work on it without having computational knowledge.

ITU-GRAM+ main functionalities are Resource Discovery, Resource Allocation, Job Submission, Job Execution and Job MonitoringComputational or data-intensive applications use the resources according to the requests from the user in order to achieve results quickly and efficiently. The resource management system must take into account not only computational resources such as the amount of available CPU, memory, data storage capacity but also starting time of job, the financial value of the resource and the efficiency of the resource in order to find these resources. Source information on the resource can change dynamically, so to make the right choices these information should be kept constantly up to date. Grid Information Service component is implemented on ITU-GRAM+ using web-service and batch job technologies in order to satisfy this update requirement of grid nature.

Job management component is used to submit, cancel and monitor jobs for execution on available resources. Users can submit their jobs using job description language (jdl) which is a high-level, user oriented language. It has lot of information about content of job. It is based on XML structure which is a standard to exchange data between systems. After submitting jobs, it is possible to monitor job steps. The resource will update the grid resource manager on critical points of the process with unique id which is provided by Grid Resource Manager in the beginning of job submission.

Resource management is used to allocate suitable resources to jobs. As in the economy, if we think that there are unlimited demands and limited number of resources, scheduling of resources is the one of the challenge problem on Grid systems. In this work resources selection is done not only the physical properties of the resources but also the distances from each other is considered. At the same time this distance varies depending on the type of job is relatively.

Job Monitoring component describes how to user can track their jobs on ITU-GRAM+. ITU-GRAM+ has capability to allow users to cancel their jobs before starting job execution. But in this case penalty will be charged on users in order to keep stable the performance of the system. To display completed and waiting jobs to user, specific web pages are implemented. It is also possible to cancel or monitor jobs using these pages.

ITU-GRAM+ includes a web application which allow users to use a standard interfaces in order to submit their jobs. The application container is the Tomcat for both web services and web application. Apache Tomcat is an open source webserver and servlet container developed by the Apache Software Foundation .Java is the language used to develop these components. Java is currently one of the most popular programming languages in use, particularly for client-server web applications. MySQL officially, but also commonly the world's most used relational and open source database management system. ITU-GRAM+ choose the MYSQL to store all the information about users, resources, jobs and reservations. Hibernate is one of the free software that facilitated the storage and retrieval of Java domain objects via Object/Relational Mapping. Java Server Pages (JSP) is a technology that helps software developers to serve dynamically generated web pages based on HTML , XML, or other document types. Apache Struts 2 is an elegant, extensible

framework for creating enterprise-ready Java web applications. Apache Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack.From the Grid Middleware point of view there are three different part of system. The Web Application which interacts with end user, the web service which is exposed to get update status of job and the batch jobs which are running at background to retrieve update resource information and submit jobs.

As a result, ITU-GRAM+ is a grid middleware which allows to users lot of capabilities to execute their jobs. The important feature of the system is that it is implemented using all open source software. In addition to this the other functionalities on the components are valuable for future work such as allocation resources considering not only availability of all resources but also the resource utilization. Besides using web service technology to ensure communication between resources and middleware is added value for Grid system. ITU-GRAM+ is not only support data or computational grids it works with fine both of them. ITU-GRAM+ helps developers to create their own middleware based on open source software and provide standard and basic interfaces to users to submit their jobs. Behind of these functionalities basic user access protocols has been developed but other security related issues are kept out of the scope of this paper. In addition to this recover mechanism for failure states is handled with basic implementation. Since the ITU-GRAM+ is free software it is open any improvement for future works.

1. GİRİŞ

Dağıtık hesaplamalar ve dağıtık sistemler uzun zamandır bilgisayar bilimlerinin popüler konularının arasında yer almaktadır. İletişim alanındaki gelişmeler, çok geniş ve hızla bağlantı imkânlarının sunulması dağıtık sistemlerin hızla yayılmasına olanak sağlamıştır. Bu çalışmada geliştirilen uygulamanın amacı teknik bilgiye sahip olmayan kullanıcılara uygulamalarını grid için özelleştirmeden çalıştırabilmeleri için standart arayüz ve protokoller sağlayarak hayatlarını kolaylaştırmak ve yazılım geliştiren kişilere açık kaynak kodlu yazılımlar sunarak kendi grid sistemlerinin altyapısını oluşturmalarını olanak sağlamaktır. Uygulama yeni teknolojiler kullanılarak ve sadece açık kaynak kodlu yazılımlar kullanarak geliştirilmiştir.

1.1 Grid Sistemler

Grid, farklı grupların sahip olduğu ve yönettiği bilgisayarlar, ağlar, veritabanları ve bilimsel araçların işbirlikçi ve toplu kullanımlarını sağlayan altyapıdır. [1] Ian Foster tarafından bir sistemin grid olabilmesi için tanımlanmış 3 madde bulunmaktadır. Bunlardan ilki, tek bir merkeze ait olmayan kaynakların yönetiminin yapılması, ikincisi açık, standart ve genel amaçlı yazılmış arayüz ve protokollerden oluşması ve son olarak da azımsanmayacak ölçüde servis kalite dağıtımının yapılıyor olması. [2] Grid hesaplaması, dağıtık hesaplamanın geliştirilmiş hali gibi düşünülebilir. Grid, kullanıcılara coğrafik ve organizasyonel olarak dağıtılmış makine, veri, insan gibi kaynaklara erişim olanağı sağlar. Organizasyonlara da var olan yazılım ve donanımların kaynaklarından kullanılmayan kısımlarını alarak grid sisteminden faydalanmalarını sağlar. Kullanıcılara tek bir makine üzerinde gerçekleştiremeyeceği hesaplama ağırlıklı işlemlerini koşturmak imkanı sağlar. [3]

1.1.1 Grid hesaplama faydaları

Birçok organizasyonda kapasitesi altında kullanılan büyük ölçüde bir kaynak potansiyeli bulunmaktadır. Birçok bilgisayar zamanın sadece % 5'lik bir zaman diliminde işlem yapıyor, geri kalan zamanda hiçbir işlem gerçekleştirilmiyor.

Göreceli olarak birçok sunucu bilgisayarların da boş zamanları bulunuyor. Grid hesaplama bu kapasite altında kullanılan kaynaklardan yararlanmak için arayüzleri sağlar.[4]

Grid sisteminin dikkat çekici özellerinden birisi de büyük ölçüdeki paralel CPU kapasitesidir. Uygulamalar, paralel çalışacak şekilde alt programcıklar olarak geliştirilirse bu parçaları aynı anda grid ortamında çalıştırarak performans arttırabilir. Burada kısıtlayıcı etken, uygulamanın sadece belirli sayıda parçalara ayrılması veya birbirinden tamamen ayrı olmaması durumda çakışan durumların oluşabilmesidir. [4]

Grid ortamında bahsedilen kaynaklar sadece CPU, veri depolama kaynakları olarak düşünülmemelidir. Grid ek olarak birçok farklı kaynaklara da erişim sağlar. Örnek olarak lisanslar, yazılımlar ya da yazıcı gibi özel donanım cihazları bu ortamda paylaşılabilir. [4]

Grid kullanan uygulamalarda grid sistem, işlerin makinelere dağıtılırken kaynakların dengeli olarak kullanılmasını sağlar. Kaynağı dengeli olarak dağıtılması iki türlü olabilir. Bekleyen bir zirve aktivitesi olduğunda bu aktivite grid ortamındaki daha boş bir makineye yönlendirilir. Eğer grid zaten tamamen kullanılır durumda ise en düşük öncelikli çalışan iş askıya alınır veya iptal edilir daha sonra tekrar çalıştırılır, bu sırada yüksek öncelikli işler için boşluk yaratılmış olur. [4]

Grid sistemindeki herhangi bir noktada bir hata meydana gelmesi durumunda, diğer bölümler aynı şekilde etkilenmez. Grid yazılımı hata meydana geldiğinde otomatik olarak verilen işi başka bir makineye yönlendirir. Aynı şekilde herhangi bir bakım durumda grid işi başka makinelere yönlendirerek herhangi bir zarar görmeden işlemi devam ettirir. [4]

1.1.2 Grid hesaplamalarında kaynak yönetimi

Grid hesaplamanın temelinde yatan ana düşünce, daha fazla hesaplama kaynağı satın almak değil, kullanılmayan hesaplama gücünü ödünç almaktır. Kaynakların sayısı arttıkça da yönetim zorlaşmaktadır. Grid sistemlerinde kaynak yönetiminin işlevi, işlerin gerekliliklerini tanımlamak, uygulamalar ile kaynakları eşleştirmek, kaynakların atamasını, zamanlamasını ayarlamak ve son olarak da süreci izlemektir. Geleneksel kaynak yönetim sistemleri, kaynaklar üzerinde tamamen kontrolleri olduğunu varsayarlar. Bu düşünce ile kaynakların verimli kullanılması için birçok mekanizma yaratılabilir. Fakat Grid sistemlerde kaynaklar dağıtık ve farklı

yönetimlerin etki alanı içerisindedir. Şekil1.1'de Grid kaynak yönetim sisteminin üst seviye bir görünümü mevcuttur. Grid sistemlerde son kullanıcı işi tanımlayan belirli kısıtlar ile birlikte işi yönetim sistemine teslim eder. Sistem kullanıcıdan gelen bu kısıtlara göre boş olan kaynakları bulur ve uygun olan kaynaklara işi gönderir.

1.1.3 Grid bilgi sistemi

Bir sistemin kendine verilen işi en iyi şekilde planlayabilmesi, elindeki kaynaklar ve iş ile ilgili ne kadar bilgi sahibi olduğu ile yakından ilintilidir. Teorikte sistemler bu bilgilere %100 oranında sahip olmak ister ve bu bilgilerin tamamın doğru olduğunu kabul eder.[6]Fakat Grid sistem yönetiminde bu şekilde kesin çizgiler ile planlamanın, yaşanılan deneyimlere bakıldığında çok da mümkün olmadığı görülmüştür. Grid yönetim sistemleri genellikle yerel kaynaklar ile ilgili bu bilgileri grid bilgi sisteminden alırlar. Bu servis aracılığı ile kaynak sistemin sahip olduğu kaynak ile ilgili birçok bilgi alınabilir. Bu bilgilerin bazıları statik ve bazıları da dinamik bilgilerdir. Örneğin kaynak üzerindeki yazılım araçları, kaynağın işletim sistemi, hangi dosya sisteminin ulaşılabilir olduğu gibi statik bilgileri kaynaklar sisteme katılırken bir defaya mahsus alınıp sistem üzerinde tutulur ve hızlı erişim sağlanabilir. Kaynağın merkezi işlem birimi kapasitesi, veri depolama aygıtların boyutu dinamik olarak değişebilen bilgilerdir ve bu bilgilerin sık güncellenmesi gerekmektedir. Kaynaklar üzerindeki bu dinamik bilgilerin standartlar üzerine kurulu bir yapı ile güncellenmesi, sisteme yeni katılan kaynakların daha kolay uyum sağlaması ve ortak bir dil ile tüm kaynaklar ile haberleşmesi önemlidir.

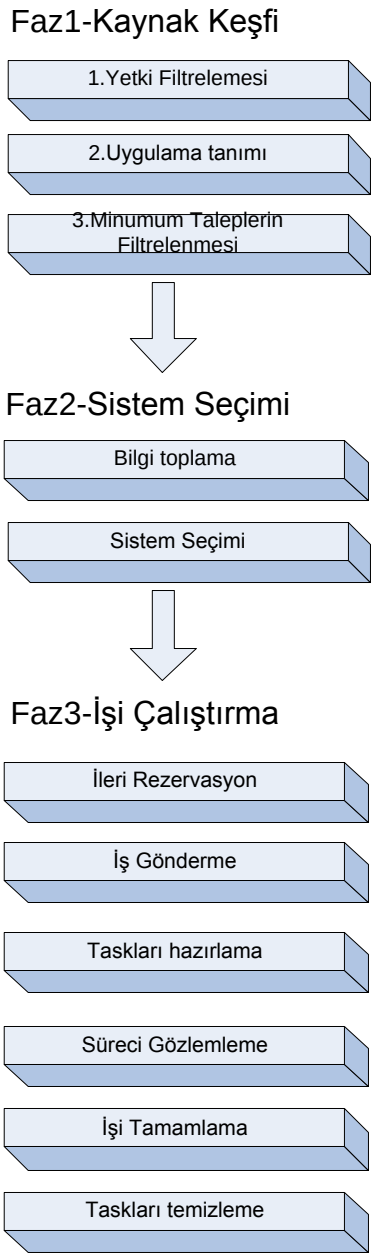
1.1.4 Grid iş sıralaması

Kaynaklar hakkında güncel ve doğru bilgiler elde edildikten sonra bu kaynaklara işlerin nasıl atanacağı ve nasıl koşturulacağı işlemleri genel olarak üç bölümde incelenebilir.

1.1.4.1 Kaynak keşfi

İlk öncelikli olarak işi gönderen kullanıcının erişim hakkı bulunduğu kaynakların hangileri olduğuna karar verilir. Bu aşama sonunda kullanıcıya ulaşabileceği makinelerin veya kaynakların listesi verilir. [6] Şu an ki GIS uygulamalarında kullanıcılar hakları olmayan birçok kaynağın da durumlarını görebilmektedir. Fakat bu kaynakların giderek arttığı bir ortamda çok da mantıklı değildir. Burada önemli

olan kullanıcı adı, şifre ve makine bilgilerinin bulunduğu dosyayı güvenli bir şekilde saklayıp, kullanıcı talebini ilettikten sonra bu liste üzerinden kullanıcının



Şekil 1.1: Grid iş sıralama.

ulaşabileceği makinelerin listesini kullanıcı ile paylaşmaktadır. Uygun kaynakları bulabilmek için kullanıcılar iş için gerekli minimum parametreleri tanımlamalıdırlar. Bu parametreler çok geniş ve işe göre değişen parametreler olabilir. Fakat hangi işletim sistemi üzerinde çalışacağı veya özel bir altyapı gerekiyorsa bunun ne olduğu gibi statik parametreler tanımlanabilir. Bunun yanında ne kadar RAM ihtiyacı

olacağı, bağlantı hızının ne olması gerektiği, ne kadar bir boş alana ihtiyacı olduğu gibi işe göre değişebilen taleplerin de kullanıcı tarafından girilmesi beklenmektedir. Kullanıcılar açısından bir standart oluşturmak amacıyla Global Grid Forum, İş Tanımlama dili (JDL) olarak belirtilen xml yapısında bir dil ortaya çıkarmışlardır.Bu xml yapısındaki dosya içerisinde iş ile ilgili en ayrıntılı detaya kadar bilgiler girilebilmektedir.[7] Kullanıcılardan beklenen en önemli bilgilerden biri de kaynağa ne kadar süre ihtiyacı olacağı konusunda bir öngörü de bulunabilmesidir. Bu öngörü her ne kadar seçilen sisteme göre değişebilmekte ve zor karar verilebilen bir parametre olmasına rağmen, işin değişkenliği ve ne tür bir işin çalıştıracağı konusunda kaynak yönetim sisteminin de herhangi bir bilgisi olamayacağından kullanıcılardan beklenilen parametreler arasındadır. Tabi burada kaynak yönetim sistemi bu bilginin yaklaşık bir değer olduğunu ve beklenilmeyen durumların oluşabileceğini de göz önünde bulundurarak davranmalıdır. Her ne kadar konuyla ilgili çok az çalışma yapılmış olsa da otomatik olarak iş ve işin ne kadar sürdüğü gibi bilgileri toplayarak bir geçmiş bilgisi oluşturup daha sonra tahmin etme amaçlı saklayan çalışmalar da vardır. Kullanıcıdan gerekli asgari ölçüde istekler alındıktan sonra bu istekleri karşılayamayacak olan kaynakların elenmesi gerekmektedir. Bu eleme genellikle statik olarak belirtilen işletim sistemi ya da altyapı gereklilikleri gibi kullanıcın hiç bir şekilde o makine üzerinde çalışamayacağı parametreler üzerinden yapılır.

1.1.4.2 Sistem seçimi

Bir önceki aşamada elenen ve işe uygun olabilecek kaynak listesi arasından en uygun iş/kaynak ikilisini seçebilmek için dinamik kaynak bilgilerine ihtiyaç duyulmaktadır. Her kaynak için ulaşılabilir olan bilgiler değişebilmektedir. Kullanıcıların bu bilgilere erişim için kullanacağı daha önceki bölümlerde belirtilmiş olan GIS bileşeni vardır. Bilgileri elde ettikten sonra grid uygulamasına göre değişen ve karar vermek için kullanılan farklı ölçütler vardır.

1.1.4.3 İşi çalıştırma

Verilen sistemi en iyi şekilde değerlendirebilmek için kaynakların hepsi ya da bir kısmı rezerve edilebilir. Kaynağa göre bu gelişmiş rezervasyon, otomatik veya insan yardımıyla kolay ya da zor bir şekilde yapılabilir. İleri bir tarih için bu rezervasyonu yapabiliyor olmak kullanıcı ile sistem arasında daha önceden belirlenen servis kalite

anlaşması yapılmış olması gerekir.Ama bu aşama her grid sisteminde olması zorunlu değildir, seçmelidir.Kaynaklar seçildikten sonra işi kaynağa gönderme aşamasına gelinmiştir. Bu işin tanımı kaynak üzerinde ufak bir komut çalıştırmak ya da daha kurulum dosyası çalıştırmak olabilir. Bu konuda belirli standartlar olmadığı için işin içeriğine göre işin kaynağa gönderilme işlemi karmaşıklaşmaktadır. Bu sebeple işi göndermeden önce kaynak üzerinde yapılması gereken görevleri belirlemek gerekmektedir. Bunlar giriş dosyasının sisteme gönderilmesi veya sistem üzerinde yetkilendirmelerin yapılması olabilir. İşin kaynak üzerinde çalıştırıldığı andan itibaren kullanıcılar işin nasıl gittiğine dair bilgilendirme almak isteyebilirler. Kaynak üzerinde bu süreç iki şekilde izlenebilir. Kaynağın kaynak yönetim sistemini belli zaman aralıkları ile güncellemesi ya da kaynak yönetim sistemin kaynağa belli zaman aralıkları ile sorgular göndererek işin nasıl devam ettiğine dair kendini güncel tutması şeklinde olabilir. İş bittikten sonra kullanıcıların işin bittiğine dair uyarılması gerekir ki bu da genellikle mail yöntemi ile olmaktadır. İş tamamlandıktan sonra oluşan çıktı dosyalarının kaynaktan silinip kaynak yönetim sistemine gönderilmesi gerekmektedir.

1.1.5 Kaynak yönetim sistemlerinin sınıflandırılması

Grid kaynak yönetim sistemi sınıflandırılması aşağıda verilmiştir. Sınıflandırmalar makinelerin organizasyonuna, kaynak isim uzayı (namespace) organizasyonuna, kaynakların bulunması, dağıtılması ve programlanmasına göre yapılmıştır.[5]

Grid sistemlerde makinelerin organizasyonu sistemin ne kadar ölçeklenebilir olduğu ile doğrudan ilişkilidir. Makine organizasyonu aynı zamanda kaynak yönetim sisteminin haberleşme desenini de etkiler. Makine organizasyonları merkezi ve merkezi olmayan şekilde ikiye ayrılır. Merkezi organizasyonda tek bir kontrol noktası ya da belirlenmiş kontroller topluluğu tüm makineler için planlama işlemini yaparlar. Diğer taraftan merkezi olmayan kontrollerde, kontrol grid sistemi içerisindeki kaynaklara dağıtılmıştır. Diğer sınıflandırma tipi de düz, hiyerarşik ve hücre yapısı organizasyonlardır. Düz organizasyonlarda tüm makineler arada başka bir makineye ihtiyaç duymadan doğrudan haberleşebilirler. Hiyerarşik organizasyonlarda aynı seviyedeki tüm makineler üstü altı ya da akranları ile doğrudan haberleşebilirler. Bu organizasyon tipi genelde grid için daha uygundur. Son olarak hücre yapısı organizasyonu, aynı hücre içindeki makineler birbirleri ile

düz organizasyon şekli ile haberleşir fakat hücre dışındaki makineler ile hücre içinde tanımlanmış özel makineler aracılığı ile haberleşebilirler.

Grid kaynak yönetim sistemi küresel olarak isimlendirilmiş kaynak havuzunu bulundurmaya ihtiyaç duyar. Kaynak isim uzayı organizasyonu kaynakların bulunması ve dağıtılmasında önemli rol oynar. İlişkisel, hiyerarşik ve çizelge tabanlı(graph based) olmak üzere 3'e ayrılır.

İlişkisel isim uzayı kaynakları kullanım ve ilişkilisel öğelere göre; hiyerarşik isim uzayı ise sistemlerin sistemi şeklinde bir yaklaşım izler ve isimleri hiyerarşik olarak aşağıya doğru dönerek oluşturur. Sonuncu yaklaşım çizelge tabanlı olan ise kaynakları birbirine bağlar ve kaynak isimlerini de bu bağlara göre oluşturur.

Kaynak bulma servisleri Grid Yönetim sisteminin en önemli fonksiyonlarından birisidir.Kaynaklardan elde edilen bu bilgiler iş sıralaması tarafından kullanılır.Kaynak bulma yaklaşımları iki sınıfta incelenebilir. Sorgu tabanlı ya da araç tabanlıdır. Sorgu tabanlı olan kaynakların durumunu kontrol etmek için merkezi ya da dağıtık bilgileri sorgular. Vekil tabanlı sistemlerde ise keşif vekilleri tarafından bilgiler toplanır. Kaynak bilgileri veritabanı üzerinde güncelleme yaklaşımı kaynak dağılma sürecini de ortaya çıkarmıştır. Bu süreç kaynakların keşfine yardımcı bir süreçtir ve kaynakların grid içerisinde bildirilmesini sağlar. Bu protokol sistemler arasında transfer edilen veri miktarını da etkiler. Periyodik ve anlık olmak üzere ikiye ayrılır. Periyodik olarak yayma modelinde bilgiler periyodik olarak güncellenir, diğerinde ise herhangi bir durum değişikliğinde anlık olarak sistem tetiklenir ve bilgiler güncellenir.

İş sıralama organizasyonlar merkezi, hiyerarşik ve merkezi olmayan şeklinde 3 grupta incelenir. İş sıralama organizasyonu kaynak yönetim sistemi yapısını ve ölçeklenebilirliğini belirler. Merkezi iş sıralama organizasyonlarda sadece tek bir iş sıralayıcı vardır ve tüm işler bu iş sıralayıcıya gönderilir. Bu tek zamanlayıcı bütün işlerden sorumludur. Bu kararın tek bir noktada yapılması genel olarak en uygun kararların verilmesini sağlar. Fakat bu noktada meydana gelebilecek hata durumlarının yönetilmesi zordur. Hiyerarşik organizasyonlar da iş sıralama denetleyicileri hiyerarşik olarak organize edilmiştir. Bu yaklaşım, merkezi organizasyonlara göre hataya dayanıklılığı daha fazladır. Merkezi olmayan yaklaşımda ise programlama kaynak isteyene ve kaynakları sağlayanlar arasında

bağımsız olarak yapılır. Bireysel programlayıcılar birbiriler ile haberleşerek karar verirler.

1.2 Problem Tanımı

Grid uygulamalarında kaynak yönetim sisteminin en zorlu problemlerinden biri sadece belirli bir işletim sisteminde veya sadece belirli yazılımlar üzerinde çalışabilen uygulamalar için uygun kaynakları nasıl bulacağı ve bu kaynakları en verimli şekilde kullanıp kullanıcıların ihtiyaçlarını doğru şekilde nasıl karşılayacağıdır. Bu yazıda önerilen sistemde platformdan bağımsız olacak şekilde web tabanlı bir arayüz üzerinden kullanıcıların ihtiyaçlarını alarak bu ihtiyaçlara uygun kaynakları bulmak belirlediği yöntemler ile kaynağı en verimli şekilde kullanılmasını sağlar.

Grid üzerindeki kaynaklar sadece grid kaynak yönetimi tarafından kullanılmayıp günlük olarak kendi işlerini de yürütecek şekilde sisteme dahil olmaktadır. Bu durumda kaynakların bu değişen dinamik yapısına uygun bir sistem kurmak ve bu dinamik yapı ile birlikte statik olan bilgileri de birlikte kullanıp uygun kaynağı bulmaya çalışmak gerekmektedir. Yeni önerilen sistem bu probleme çözüm olarak kaynak üzerinde bu dinamik bilgileri alabileceği bir servis oluşturulmuş ve gerektiği zaman bu servisi çağırarak kaynağın güncel bilgilerine ulaşabilmiştir. Fakat bu problem aslında rezervasyon yapılan sistemler ve veri depolama olarak kullanılacak kaynaklar düşünüldüğünde yeni bir probleme yol açılıyor. Rezervasyon anında kaynağın o andaki bilgisine göre rezervasyon işlemi yapılır. Fakat daha öncesinde aynı kaynak için yapılmış bir rezervasyon işlemi var ise ve bu işlem sonucu kaynak üzerindeki veri depolama ve dosya sistemlerindeki indeks bilgileri değişmiş ise bir sonraki iş bundan haberdar olmayacağı için istenilen sonuca ulaşılamayacaktır.Bu problemi çözmek için istenilen kaynak üzerindeki işlemde yazma işlemi bulunuyor ise bunu ayrıca işaretleyip bir sonraki listelemelerde bunu belirtmek gerekir. Diğer problemlerden biri kullanıcıdan yapılacak iş ile ilgili bilgileri alırken, uygulamanın çalıştırılabilmesi için gerekli asgari kaynak gerekliliklerini de standart bir arayüz ve format ile alınmasıdır. Kullanıcının yapılmasını istediği işin sonuçları dışında, arka tarafta gerçekleşen işlemlerin teknik ayrıntılarından haberdar olmaması gerekmektedir. Bu probleme çözüm olabilmesi için bir web uygulaması tasarlanmış

ve kullanıcılara belirli arayüzler ve standartlar sağlanarak iş taleplerini girmeleri istenmiştir.

Aynı şekilde sisteme dahil olan kaynakların da kaynak yönetim sistemi ile standart arayüz ve uygulamalarla haberleşmesi sistemin geniş kapsamda kullanılabilirliği açısından önemlidir.Bu standartlara ulaşmak en önemli problemlerden biri olmak ile birlikte günümüz yazılım teknolojilerinden yararlanıp sistemin bu teknolojiler üzerinde kurarak büyük ölçüde bu probleme çözüm sağlanabilir. Bu tez içerisinde önerilen sistemde kaynaklar üzerinde standart web servisler tanımlanmış ve sunucu tarafında bu standart web servisleri çağıracak istemciler yaratılmıştır.

Kaynakların verimli kullanılması ve kullanıcıların bu kaynaklardan en adaletli şekilde yararlanabilmesi de grid kaynak yönetim sisteminin sorumluluklarından biridir. Bu durumu en iyi şekilde yönetebilmek için işleri uygun kaynaklara atarken kaynakların niteliği, sistemdeki iş yükü, rezervasyonun işin çalıştırma tarihinden ne kadar önce verildiği ve kullanıcının ne kadar sıklıkta uygulamadan yararlandığı gibi ölçütler göz önüne alınmalı ve bu ölçütler doğrultusunda kullanıcının da bütçesine uygun farklı seçenekler sunulmalıdır.

Grid yönetim sisteminin karşılaşmış olduğu problemlerden biri de işin kaynağa gönderildikten sonra işin başlayabilmesi için gerekli giriş verileri ve uygulama kodlarının kaynak üzerine taşınması sırasında geçen süredir. Bu süre boyunca kaynak üzerinde herhangi bir işlem başlatılmadığı için kaynak boş yere beklemektedir. Bu durum, işin çalışma süresinin içerisinde hesaplanan bir parametre olmadığı için de kullanıcılar tarafından doğru öngörüler de bulunması zorlaşmıştır. Bu kaybedilen süreyi önlemek amacıyla işlem kaynak sisteme gönderilmede önce işlem için gereken dosyalar kaynak sisteme gönderilir böylece iş başlamadan önce iş için gerekli tüm dosyalar kaynak sistem üzerinde hazır olarak beklemektedir. Önerilen kaynak yönetim sisteminde tüm bu problemler ele alınmış ve çözüm için gerekli adımlar atılmıştır.

1.3 Tezin Bölümleri

İlk bölümde grid sistem ile ilgili genel bilgiler verilmiştir. Grid sisteminin bölümlerinden biri olan kaynak yönetim sisteminin hangi bileşenleri içereceği ve her birinin grid sistem üzerindeki görevleri anlatılmıştır. Aynı bölüm içerisinde grid

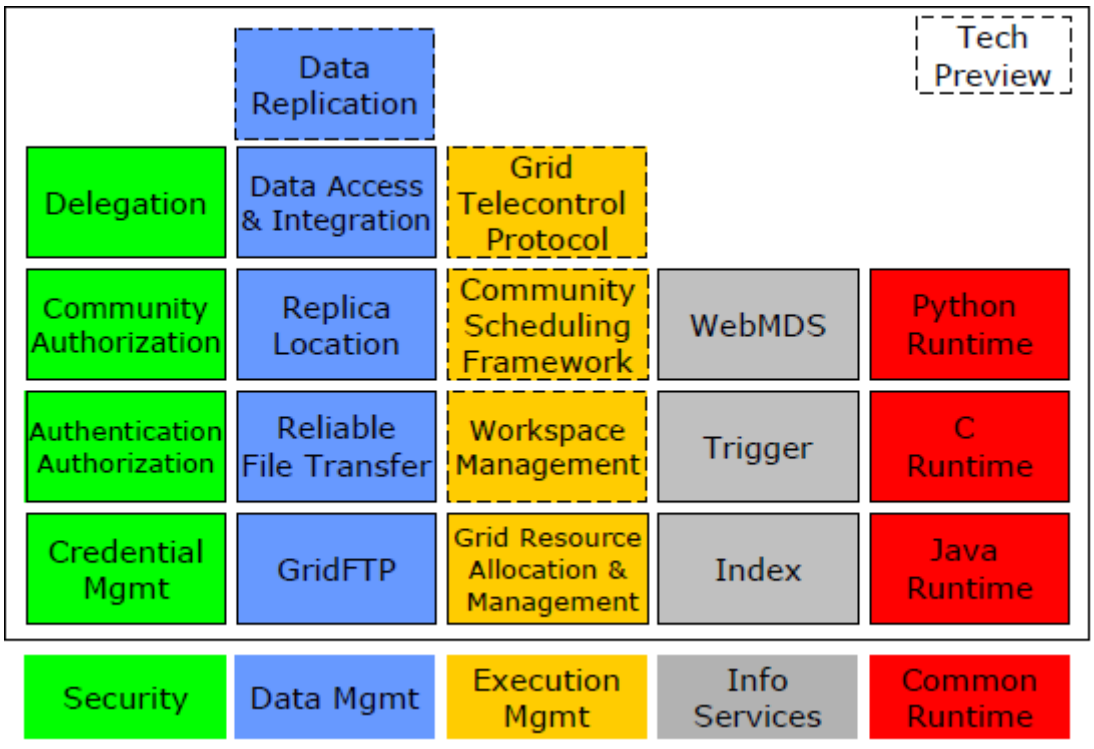
sisteminde var olan problemler ele alınmış ve bu çalışma kapsamında hangilerine nasıl çözümler getirildiği anlatılmıştır. İkinci bölümde var olan grid sistemlerin özellikleri anlatılmıştır. Bu proje kapsamında önerilen uygulama ile farkları ve benzerlikleri gösterilmiş ayrıca bu çalışma esnasında motivasyon kaynaklarının neler olduğundan bahsedilmiştir. Üçüncü bölümde geliştiren uygulamanın hangi teknolojiler kullanılarak geliştirildiği ve neden bu yazılımların seçildiği ile ilgili bilgiler verilmiştir. Dördüncü bölümde, ITU-GRAM+ sisteminin tasarımı UML diyagramları oluşturularak yapılmış ve mimarisi oluşturulmuştur. Beşinci bölümde tasarımı yapılan ITU-GRAM+'in nasıl geliştirildiği ve gerçeklendiği ,algoritmaların neler olduğu anlatılmıştır. Son bölümde ise yapılan çalışmanın sonuçları ve gelişime açık yönlerinin neler olduğu anlatılmıştır.

2. LİTERATÜR ARAŞTIRMASI ve MOTİVASYON

Grid sistemler, servis, veri ve hesaplama olarak 3 kategoride incelenebilir. Bu bölümde var olan farklı grid sistemler incelenecek ve ITU-GRAM+ ile karşılaştırılacaktır.

2.1 Grid Sistem Uygulamaları

Globus yazılımı dağıtık kaynakları kullanıcılar arasında tamamen yetkiyi ele almadan paylaşmalarına olanak sağlayan uygulamaları tasarlamaya imkan vermektedir. Bu geliştirme ortamı, kaynakları izleme, keşfetme ve yönetmenin yanı sıra güvenlik ve dosya yönetim işlemleri yapan birçok servis ve kütüphaneleri içerir. Globus içerisinde kaynak yönetiminden sorumlu parça, dağıtık kaynaklarına istekte bulunma ve kullanmak için tek ve yaygın protokoller ve arayüzler sağlamaktadır.
Globus Toolkit 4 sürümünde işi uzaktan çalıştırmak için GRAM servisi kullanılmaktadır.



Şekil 2.1 : Globus Toolkit yapısı

GRAM, işleri(iş tanımlama dili ile tanımlanmış) çalıştırmak için istekleri gönderen, iş sonuçları izleyen ve kontrol eden mekanizmaları içerir.[7,10]

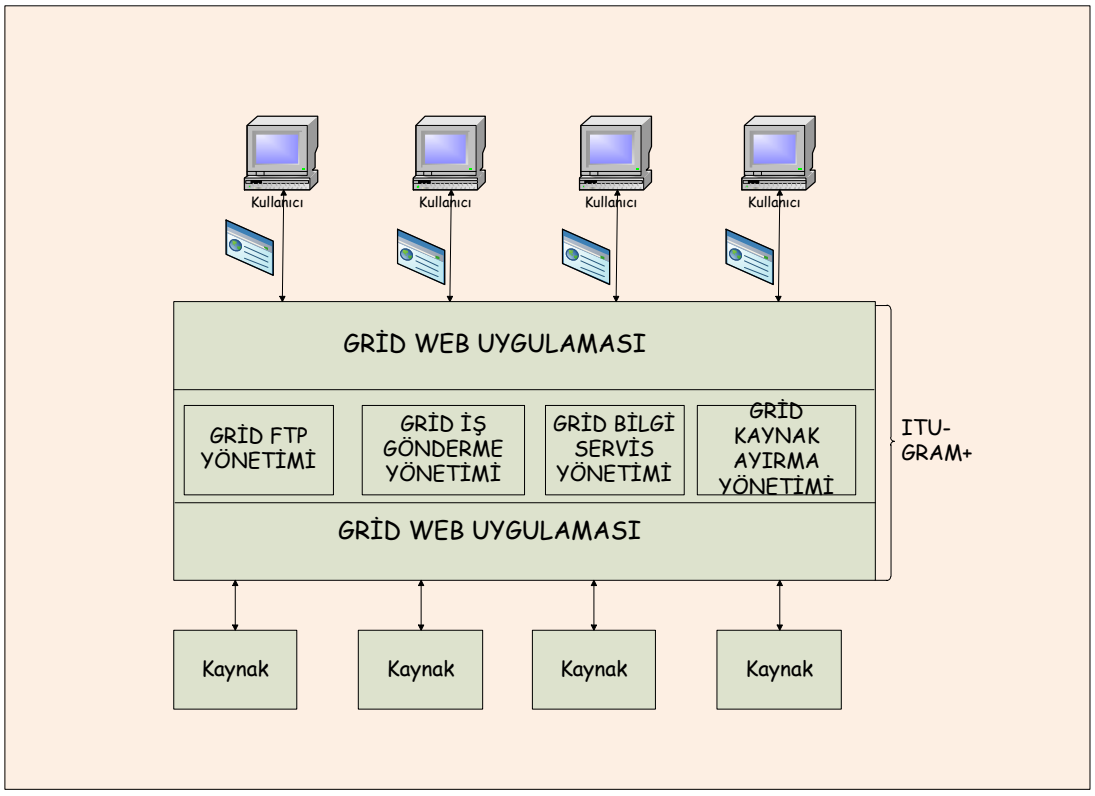
Şekil 2.1'de Globus Toolkit 4 sürümüne ait tüm bileşenleri içeren bir grid altyapısı gösterilmiştir. Bunlardan bu çalışma kapsamında da ele alınan başlıca bileşenlerin kısa tanımlamaları aşağıda verilmiştir.

GRAM: Grid kaynak tahsis ve yönetim protokolü, bir işlem isteğinin uzaktaki bir kaynağa atanmasını ve hesaplamanın gözlemlenmesini düzenler.

GRIDFTP:Globus GridFTP sunucu ve istemci araç ve kütüphaneleri ile standart FTP protokolüne göre daha güvenilir, daha hızlı ve büyük miktarlarda veri taşımak için tasarlanmış bir üründür.

MDS-Index: Grid kaynaklarının durumu ve yapıları hakkında bilgilerin alınmasıyla ilgili işlemleri düzenler.

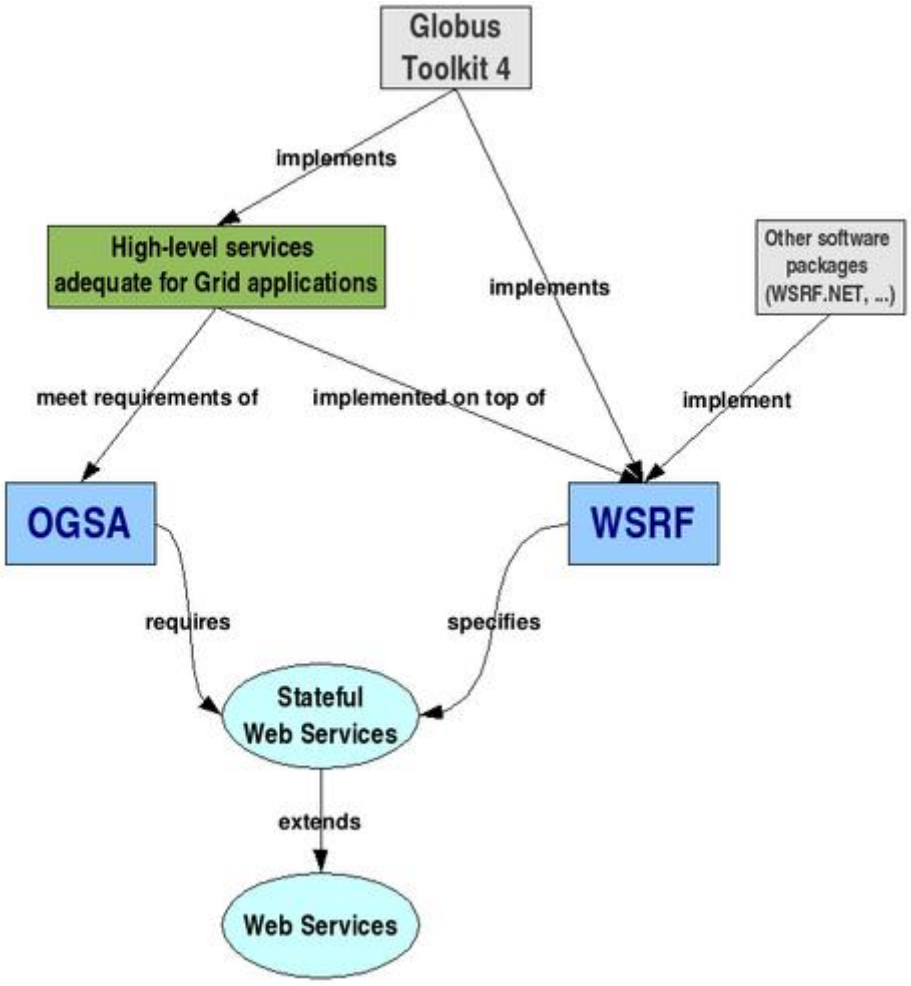
GSI: Grid güvenlik altyapısı (Grid Security Infrastructure) açık anahtar şifreleme yöntemiyle kimlik denetleme ve yetkilendirme işlemlerini düzenleyen protokoldür



Şekil 2.2 : ITU-GRAM+ bileşenleri

Şekil 2.2'de ITU-GRAM+ üzerinde yer alan bileşenler gösterilmiştir. Uygulamaya ITU-GRAM+ isminin verilmesinin sebebi, ana fonksiyonu olan kaynak rezervasyon

ve yönetiminin yanında SFTP protokolü ile veri taşınmasını sağlaması ve kaynaklar üzerinde koşan web servisler ile de kaynaklar hakkındaki güncel bilgilere ulaşılması ve iş ile ilgili izleme süreçlerini içermesidir. OGSA, Globus Grid Forum tarafından grid tabanlı uygulamalar için ortak bir standart ve açık mimari oluşturmak amacıyla geliştirilmiştir. Her bir grid uygulaması için yukarıda belirttiğimiz kaynak yönetimi, iş yönetimi gibi servislerin olması ve bunların birbirleriyle haberleşmesi gerekmektedir. Her bir grid uygulama yazan kişinin, bunları farklı standart ve arayüzlere göre geliştirmesi ve her bir yazılım parçasının birbirleriyle haberleşmesini sağlamaya çalışması tam bir karışıklığa yol açar. Bu standartları oluşturmak amacıyla dağıtık sistemlerde kullanmak için teknoloji olarak web servis teknolojisi seçilmiştir. Fakat web servislerin *stateless* bir yapıya sahip olmasından dolayı grid ihtiyaçlarını tam karşılayamamaktadır. Bu nedenle web servisleri *stateful* hale getirmek için WSRF standardı ortaya atılmıştır.[8] Globus Toolkit 4,OGSA ve WSRF arasındaki ilişki Şekil 2.3'de gösterilmiştir.[9]



Şekil 2.3 : GT4,OGSA, WSRF, Web Service arasındaki ilişki

Globus'a alternatif olarak Legion ve CondorG uygulamaları bulunmaktadır.
Legion Grid Portal, grid sisteme arayüzler sağlar. Kullanıcılar bu arayüzler sayesinde dosyaları görüntüleyebilir, işleri gönderebilir ve iş sonuçlarını takip edebilir. Virginia Üniversitesinde geliştirilen bu sistem MPI ve PVM de yazılan kodları destekleyerek kullanıcılarına büyük ölçekli ve karmaşık kaynak havuzlarından yararlanma imkanı sunar.[11]
Condor-G, Condor ve Globus projelerinin birleştirilmesi sonucu ortaya çıkan bir grid sistemidir. Ağ üzerindeki atıl kaynakları keşfederek ve bu kaynaklara uygun uygulama görevleri atayarak yüksek verimli hesaplamaları desteklemek için tasarlanmıştır.[12,13]

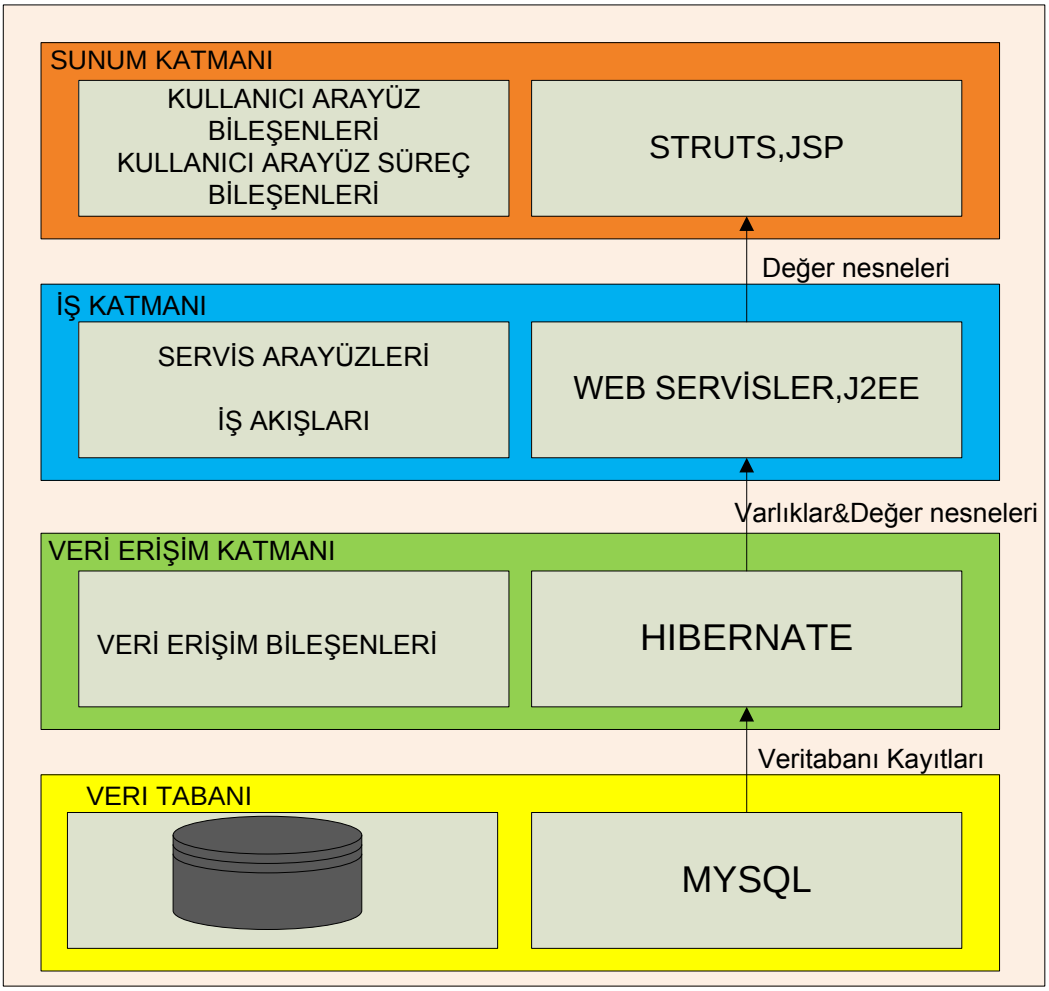
2.2 Motivasyon

Birçok grid portalı, bilim adamları ve araştırmacılar için kendi programlarını uzaktan gönderme, giriş ve çıkış dosyalarını saklama ve kaynaklar ile ilgili bilgileri edinmek için özelleştirilmiş arayüzler sağlar. Bunlardan bazıları çok geniş kitleler tarafından kabul edilmiş ve grid sistem için tüm gereken bileşenleri barındırır. Fakat benim bildiğim uygulamalar arasında sadece açık kaynak kodlu yazılımlar kullanılarak geliştirilmiş bir grid ara katmanı bulunmamaktadır. Yazılım geliştiriciler bakış açısıyla bakıldığında çok basit bir şekilde kendi grid ara katmanlarını oluşturabilir ve kendileri de yeni özellikler ekleyerek bunu geliştirebilir. Günlük bilgisayar kullanıcıları, bilim adamları ve iş amaçlı kullanan kullanıcılar bakış açısıyla bakıldığında herhangi bir bilgisayar dalında altyapısı olmadan kolay bir şekilde kullanmaya başlayabilir. Bu iki önemli nokta çalışmam süresinden beni motive eden etkenler arasında yer almaktadır.

3. ITU-GRAM+ MİMARİSİ

Bu bölümde kaynak yönetim sistemini tasarlamak ve gerçeklemek için kullanılan teknolojiler tanıtılmıştır. Şekil 3.1'de kullanılan teknolojiler kullanım amaçlarına göre katmanlı yapıda gösterilmiştir. Tüm katmanlar doğrudan bir üst veya bir alt kammanındaki bileşenler ile bilgi alışverişinde bulunur.

İleriki bölümlerde her bir teknoloji tek tek incelenecek ve neden bu teknolojilerin tercih edildiği ile ilgili bilgiler verilecektir.



Şekil 3.1 : Kullanılan teknolojiler.

3.1 Mysql

Mysql, çoklu iş parçacıklı, çok kullanıcılı hızlı ve sağlam bir veritabanı yönetim sistemidir. UNIX, OS/2 ve Windows platformları için ücretsiz dağıtılmakla birlikte ticari lisans kullanmak isteyenler için de ücretli bir lisans seçeneği de mevcuttur. Kaynak kodu açık olan MySQL'in pek çok platform için çalıştırılabilir ikilik kod halindeki indirilebilir sürümleri de mevcuttur. Ayrıca ODBC sürücüleri de bulunduğu için birçok geliştirme platformunda rahatlıkla kullanılabilir. Web sunucularında en çok kullanılan veritabanıdır. MySQL için çok çeşitli grafik arayüze sahip programlar mevcuttur. Bunlar içerisinden en bilineni yine MySQL'i geliştiren firma tarafından geliştirilmiş ücretsiz bir yazılım olan MySQL GUI Tools'dur. MySQL, tuttuğu tablolarla çok kullanıcılı sistemlerde söz konusu olan erişim hakları sorununu başarılı bir şekilde çözmektedir. MySQL'in 4.0 sürümü ile birlikte *transaction* desteği, 4.1 sürümüyle birlikte de alt sorgu desteği eklenmiştir. Ayrıca veri tutarlılığını sağlama işinin programcıya bırakılması tercih edilmiştir, ancak bu bir dezavantaj olarak görülmeyebilir. Çünkü pek çok veritabanı programcısı VTYS'lerdeki veri tutarlılığının esnek olmayan, zorlayıcı bir özellik olduğunu düşünmektedir. Bu tez içerisinde MySQL'in tercih edilme sebeplerinden en önemlisi açık kaynak kodlu ücretsiz bir veritabanı olması ve bu sayede herhangi bir lisans ücreti ödemeden yazılan uygulamayı bu veritabanı üzerinde çalıştırılabilmesi. Ayrıca kurulumunun basit ve hızlı bir şekilde yapılabilmesi projeyi geliştirirken hızlı yol alınmasına yardımcı olmuştur.[14]

3.2 Java

Java, Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kodlu, nesneye yönelik, zeminden bağımsız, yüksek verimli, çok işlevli, yüksek seviye, adım adım işletilen bir dildir. JVM yorumlayıcısına sahip herhangi bir platforma taşınabilir. Ağ ortamı ve yazılımın platform bağımsız olarak çalışması(değişik bilgisayar türlerinden ve değişik işletim sistemlerinde çalışması) düşünülerek geliştirilmiştir. Bu nedenle diğer dillerden farklı olarak aynı zamanda kendiside bir platformdur. Java'nın diğer önemli özelliği nesneye yönelik bir dil olmasıdır. Nesneye yönelik diller, nesneleri gerçek dünyadakine benzer bir yapıda tanımlayarak anlaşılmalarını kolaylaştırır. Nesneleri gerçek dünyadaki gibi bir masa, sandalye, bilgisayar, ısı esanjörü gibi tanımlayarak programlamak insan beyninin

anlaması açısından çok daha kolaydır. C++ nesnel kökenli programlama yapabilen bir dildir. Fakat yapısal bir programlama dili olan C dilinin bir uzantısı olarak geliştirildiğinden tam anlamıyla nesneye yönelik bir dil olduğu söylenemez. Java dilini geleceğin dili yapan diğer bir özelliği de çok kullanımlı ve paralel kullanımlı bir dil olmasıdır. Çok kullanımlılık birden fazla işlemin ayni anda yapılabilmesinin tanımıdır. Paralel kullanımlılık ise birden fazla programın aynı anda belleği beraber kullanabilmesidir. Örnek olarak Word ve Excel programlarının Windows NT ortamında aynı anda kullanılmasıdır. ADA gibi bazı eski program dillerinde çok kullanımlılık programlanabiliyordu. Paralel kullanım olanaklarını sunan ilk bilgisayar dili ise Java'dır. Paralel kullanım, paralel programlama kavramından ayrıdır ve karıştırılmamalıdır. Paralel programlamada birden fazla Bilgi işlem Ünitesine (CPU) ayrı programlar veya bir programın ayrı parçaları gönderilir. [12] CPU'nun kullanım zamanı küçük parçalara ayrılarak değişik Program veya Program parçacıkları bu zaman paketçiklerini paylaşarak kullanırlar. Java'yı önemli bir Program dili haline getiren en önemli özelliği ise kullanılan bilgisayardan bağımsız olmasıdır. Java'da yazılan bir Program Unix, Macintosh, Windows 95 veya Windows NT veya herhangi bir 32 bit makinede hiç değiştirilmeden kullanılabilir. Java programlarının grafikleri "World Wide Web" sayfalarının programlama dili olan html (hypertext markup language) ile aktarılır. Bu yüzden html ve java programlarını birlikte kullanmak ve java programlarını gerçek zamanda www sayfalarında göstermek mümkündür. Kendi web sayfanızı veya web'de yazılmış kitabınızı bütün dünyaya aktarırken yaptığınız analizleri de bu kitabın dinamik bir parçası olarak sunabilirsiniz. Java uygulamalarınızı yazmak için, Notepad gibi bir program dışında özel olarak geliştirilmiş görsel arayüze sahip olan programlar da vardır. Bu tür programlarda, kodları hem yazar, hem derler hem de çalıştırabilirsiniz. Örnek olarak; Eclipse Platform, Java Builder, Net Bean, Visual Age. Java EE ise Java'nın kurumsal uygulamalar için gerekli bileşenleri içerir. J2EE, standart ve modüler bileşenlerden oluşan bir yapının karmaşık işlemler yapmadan, otomatik olarak geliştirilebilmesini sağlar. Çok sayıda makine, sunucu, veritabanı ve uygulamadan oluşan bir ortamın uyumlu çalışması için bir alt yapı oluşturur. Java Server Pages web programlama teknolojileri içerir. İçerisinde birçok teknolojiyi içermekle birlikte açık kaynaklı bir programlama dili olduğundan birçok çatı(framework) bulunmaktadır. Bu uygulama çatıları sayesinde kaliteli ve hızlı bir geliştirme ortamı sağlamaktadır. [15]

3.3 Hibernate

Hibernate, Java platformunda yazılmış bir ORM aracıdır. ORM, nesne odaklı dillerdeki nesnelerin, ilişkisel veritabanlarındaki kayıtlara nasıl karşılık geldiğini yürüten bir teknolojidir. Hibernate gibi ORM araçlarıyla, bir nesneyi veritabanına kaydetmek, yeni halini güncellemek ve sorgulama yapmak düz SQL bağlantılarına göre çok kolaydır. Hibernate gibi ORM araçlarının en önemli faydası, kod yazımını kısaltmak veya kolaylaştırmaktan öte, yazılım bakımını kolaylaştırmasıdır. Veritabanı temelli uygulamalarda, kodun 1/3´ü veritabanı erişimine yöneliktir. Veritabanındaki bir kolonunun tipinin değişmesi, yeni bir kolon eklenmesi gibi değişiklikler, bütün veri erişim kodunu tekrar gözden geçirmeyi gerektirir. Hibernate ile bu gözden geçirmeden çok yüksek oranda tasarruf edilir. Hibernate kullanılan yazılımlarda, veritabanındaki değişikliklerde yapılması gereken sadece nesnelerle tabloların birbirine nasıl eşleştirildiğinin gözden geçirilmesidir.[16]

3.4 Struts2

Struts 2, Java web uygulamalarını geliştirmek için tasarlanmış açık kaynak kodlu ücretsiz bir çatıdır. MVC tasarım şablonunu uygulayan modern bir altyapı sağlar. MVC, iş mantığı ve sunum tabakasını birbirinden ayıran bir tasarım şablonudur. Tabi bu tabakalara bir de Yönlendirici tabakasını eklemek gerekir. MVC'nin yönlendirici tabakası bir kavşakta duran trafik polisi gibidir. Nasıl trafik polisi gelen geçen arabaların hangi yöne gideceğine karar veriyorsa, yönlendirici de istemciden gelen isteklerin hangi aksiyona ulaşacağına karar verir. MVC'nin asıl çalışma prensibi projemizde kullandığımız yapıları katmanlamaktır. Veri tabanı için bir katman, sorgularımız için ayrı bir katman ve son olarak son kullanıcıya sunulacak olan ekran için ise ayrı bir katman oluşturmaktır. Model genellikle veri tabanı işlemlerimizi yani işlerimizi yaptığımız yapıdır. Veri tabanımız üzerinde yapılabilecek sorgularımızı burada belirler ve denetleyiciye atamamızı sağlarız. Bu sayede veri tabanımıza dışarıdan daha kolay erişebilir ve çeşitli yöntemlerle daha kolay idare/müdahale edilebilir hale getirir. Deneyleticiler, model yardımıyla oluşturduğumuz sorguların kullanıcı tarafından alınan veriler ile birleşip uyumlu bir biçimde çalışmasını sağlayan yapıdır. En genel tanımı uygulamaya gelen sorgulara karşı nasıl cevap verilmesi gerektiği konusunda planı yapan, hareketi planlayan bölümüdür. Bu bölüm

kullanıcıya cevap olarak döndürülen arayüzdeki statik ya da dinamik bileşenleri (html,jsp,css,...) barındırır. Diğer yardımcı yapılar yardımıyla programcılara da oldukça anlaşılır ve bütünü ile sayfa üzerinde kullanıcıya gösterilen form araçlarının bulunduğu kod parçası kalır. MVC yardımı ile karmaşık görünecek bir yapının ne kadar basitleştiğini ve sadece bizim tarafımızdan değil, yerimize devam edecek kişiler için bile çok rahat kullanılabilecek ve anlaşılır olabilecek bir yapı bırakmış oluruz. [17]

3.5 Apache Tomcat ve Apache Axis2

Apache Tomcat, Apache Yazılım Vakfı tarafından geliştirilmiş bir www sunucusudur. Tomcat, Java Servlet ve Java Server Pages teknolojileri için resmi kodlama referansı olarak kullanılan açık kaynak kodlu Servlet barındırıcısıdır. Web sayfalarının içeriğinin dinamik olmasına olanak sağlar. Gelen http isteklerini dışarıdaki bir uygulamaya aktarmaya ve sonucunu dönmek için kullanılır.[18]

Apache Axis, açık kaynak kodlu XML tabanlı web servis çatısıdır. Java ve C programlama dillerini desteklemektedir. Web servisleri oluşturmak ve dağıtmak için birçok arayüz sağlamaktadır. Apache Software Foundation tarafından geliştirilmiştir. Asıl proje de kullanılan Axis2, Axis ‘e göre daha esnek, performanslı ve daha yönetilebilir bir yapı sağlar. Proje de tercih edilmesinin en büyük sebeplerinden biri açık kaynak kodlu olması ve hem istemci hem de sunucu tarafında web servis oluşturma ve çağırmada kullanım kolaylığıdır.[19]

3.6 Kullanılan Diğer Teknolojiler

Ant, Apache Yazılım tarafından geliştirilmiş, açık kaynak Java tabanlı bir yazılım geliştirme aracıdır. Ant kısaltmasının açılımı Another Nice Tool olarak bilinir. Tomcat, JDOM gibi açık kaynak ürünlerin geliştirilmesinde kullanılmıştır. Ant build kütüğü XML formatındadır. Build kütüğünün üzerinde işlem yapabilmek için, basit bir metin editörü yeterli olacaktır. Harici bir XML Parser’a gerek yoktur. Zaten Ant’ın kurulumu sırasında JAXP Uyumlu XML Parser yüklenir.[20] Log4j, Java için geliştirilmiş en popüler günlük tutma paketidir. Log4j açık kaynak kod projesi kapsamında geliştirilmiştir. Uygulama geliştiricilerine, günlük (log) deyimlerini istedikleri boyutlarda, istedikleri biçimlerde çıktı olarak alma imkanı sağlar. Harici

yapılandırma dosyaları aracılığıyla çalışma zamanında tamamen yapılandırılabilir. Her şeyden öte, Log4j kendi kendine öğrenebilir. Log4J, hata ve bilgilendirme mesajlarımızı yaratmamızı sağlayan arayüzler içeriyor. Bu ara yüzler mesaj gereken her yerden çağrılabilir. Önemli hatalar için error(), bilgilendirme için info() ve uyarı mesajları için warn() kullanabilir.[21]

Quartz Scheduler, OpenSymphony açık kaynak topluluğu altında geliştirilen bir projedir. Özellikle kurumsal uygulamalarda daha fazla öne çıkan ve arka planda çalışacak işleri zaman planına göre başlatan ve gözlemleyen sistemlerdir. Zamanlayıcı beklenmedik şekilde kapanır veya kapatılırsa, tekrar açıldığında kaldığı yerden devam edebilir. Bu da en çok tercih edilmesinin sebeplerinden biridir.[22]

CSS, HTML'e ek olarak metin ve format biçimlendirme alanında fazladan olanaklar sunan bir Web teknolojisidir. İnternet sayfaları için genel geçer şablonlar hazırlama olanağı verdiği gibi, bağımsız olarak harflerin stilini, yani renk, yazı tipi, büyüklük gibi özelliklerini değiştirmek için de kullanılabilir. Bu tekniğin en önemli özelliği kullanımındaki esnekliktir.[23]

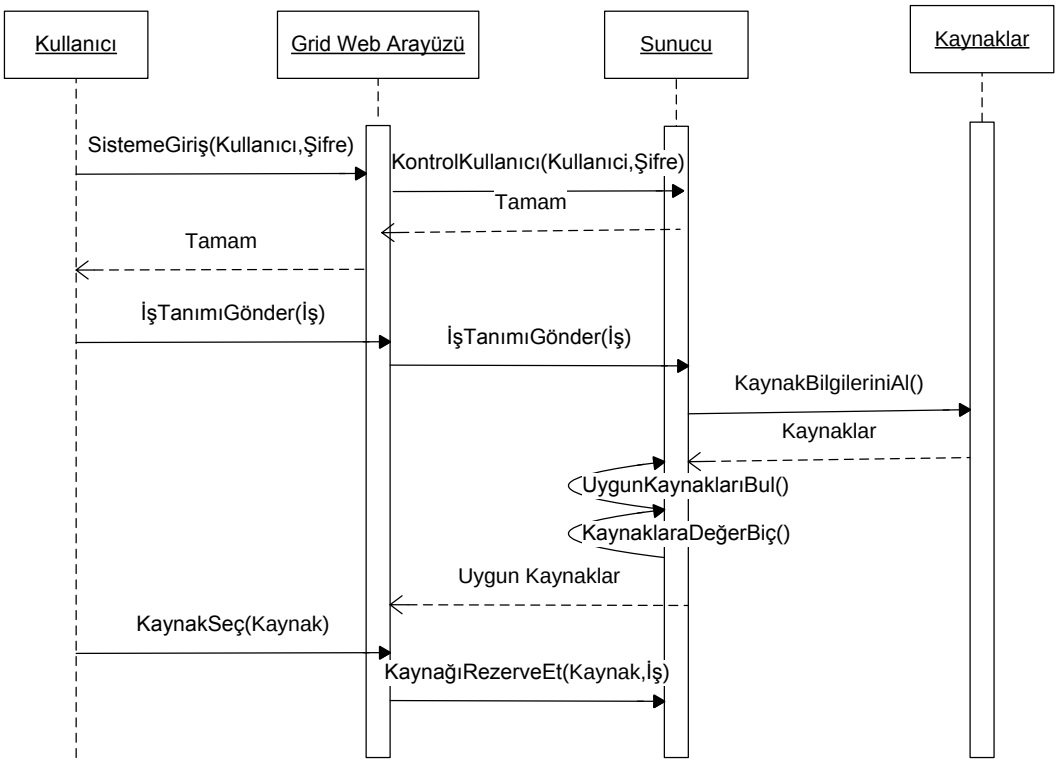
JavaScript,Netscape Navigator 2.0 ile birlikte Brendan Eich tarafından geliştirilen ve önceleri Mocha daha sonra LiveScript olarak adlandırılan ve en sonunda şu anki adını alan JavaScript dili başlangıçta sadece istemci taraflı (client-side) yorumlanan bir betik dilidir.[24]

UML, bir programlama (ya da yazılım geliştirme) dili. Daha çok yazılım geliştiriciler tarafından kullanılır. Farklı programlama dilleri kendi içinde geliştirdiği bir takım komut argümanın programcı tarafında iyi seviyede ezberlenmesi gerektiği için olası yeni teknolojik ihtiyaçları karşılama sürecinde programcının yeniden kendini geliştirme adıyla bir eğitim sürecine dahil olması kaçınılmazdır. Bu sürecin maddi kayıbını önlemek amaçlı düz doğrusal anlaşılabilinir seviyede şablonların oluşturulması hem kullanıcı insan arabirimi açısından ileri seviye programcı olmaya gerek kalmaz hem de tekrar eden farklı programlama kod arabirimleriyle uğraşılmak zorunda kalınmaması olası iş yükünü oldukça düşürmektedir.[25]

4. ITU-GRAM+ TASARIMI

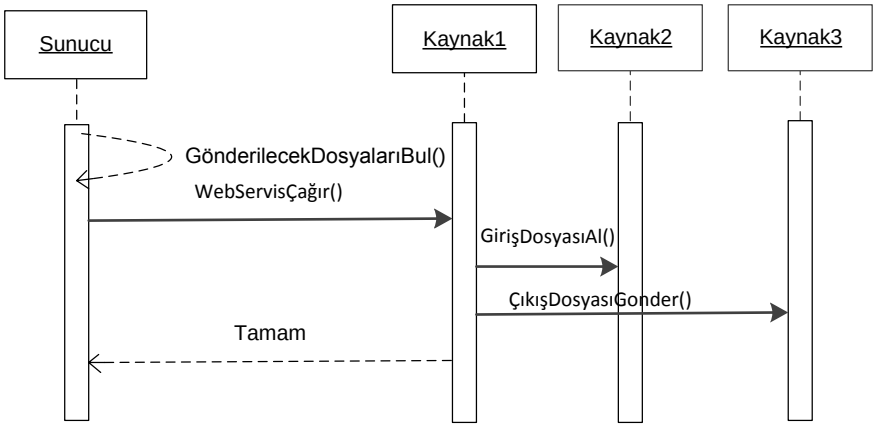
Bu bölümde, uygulama adımına geçmeden önce yukarıda bahsedilen teknolojiler ile kaynak yönetim sistemin tasarımının nasıl yapıldığı anlatılmıştır. Tasarıma başlamadan önce grid kaynak yönetim sistemin genel olarak hangi taleplere yanıt vermesi gerektiği incelenmiştir. Hesaplama veya veri yoğunluklu uygulamalar, hızlı ve verimli bir şekilde sonuca ulaşabilmek için kullanıcıdan gelen taleplere göre hesaplama veya veri kaynakları kullanırlar. Bu kaynakların bulunabilmesi için kaynak yönetim sistemi, kullanılabilir CPU miktarı, bellek, veri saklama kapasitesi gibi sadece hesaplama ile ilgili kaynakları değil aynı zamanda uygulamanın ne zaman çalıştırılacağı, finansal olarak kaynağın değerini ve o uygulamanın o kaynak üzerindeki verimini de dikkate almalıdır. Bunun yanında o kaynağın bulunduğu makinenin hata yüzdelerini, ağ transfer hızı ya da güvenlik duvarı ilkeleri gibi birçok etken daha dikkate alınabilir. Veri ağırlıklı uygulamalar, aslında yapılacak işlem basit olmasına rağmen uygulama içerisine giren ve çıkan dosyaların miktarının büyük olduğu durumlarda karşımıza çıkar. Makinelerin seçimi bu gibi durumlarda giriş verisinin bulunduğu makinede seçilirse ağ üzerinde geçen zaman azalır, eğer giriş verisi kaynak yönetim sistemi tarafından kaynağa iletilecekse bu süre zamanlama yapılırken de hesaba katılmalıdır. Birçok sistem kullanıcılardan bu verileri saklamaları için ikinci bir veri depolama aygıtları kullanmalarını önerirler fakat ideal olarak kullanıcıların bu tür endişelere kapılmamaları ve sistemin bu dosyaları istedikleri kadar saklayabileceklerini düşünmeleridir. Bu genel talepleri inceledikten sonra sistemde olabilecek tüm senaryolar UML diyagramları oluşturarak belirlenmiş ve daha sonra sistemin olması gereken parçaları ortaya çıkarılmış ve bu parçaların birbirleri ile olan ilişkileri gösterilmiştir.

Senaryo1:Sisteme daha önceden kaydı yapılmış olan kullanıcı sisteme giriş yapar. Kullanıcılardan iş ile ilgili ayrıntılar web arayüzünden alınır. Sistem uygun kaynakları niteliklerine göre değer biçtikten sonra kullanıcı ile paylaşır. Kullanıcı kendine uygun olan kaynakları seçtikten sonra sistem o kaynakları kullanıcı için rezerve eder.



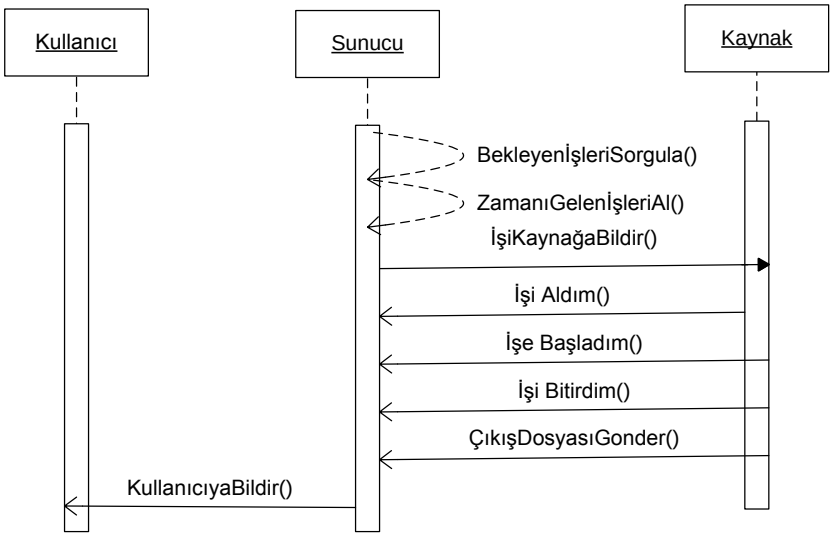
Şekil 4.1 : Senaryo1

Senaryo 2:Sistem belirli zaman aralıklarında kaynaklara gönderilmeye bekleyen işleri kontrol eder ve zamanı yaklaşan işler için uygulama kodlarını, rezerve edilen kaynaklardan işlemi yapacak olana gönderir. Bu kaynak ilgili diğer kaynaklar ile iletişime geçerek gerekli giriş ve çıkış dosyalarını alır.



Şekil 4.2 : Senaryo2.

Senaryo 3:Sistem belirli zaman aralıkları ile kaynağa gönderilmeye bekleyen işleri kontrol eder ve zamanı gelen işleri kaynağa bildirir. Kaynak işi aldıktan sonra sunucuya işin süreçleri ile ilgili bilgilendirme yapar.



Şekil 4.3 : Senaryo3.

Yukarıdaki belirtilen UML diyagramlarını gerçeklemek üzere kullanılacak olan teknolojilerin birbirleri ile ilişkileri gösterilecek ve ardından bu diyagramların bu altyapı üzerinde ne şekilde geliştirileceği anlatılacaktır.

4.1 Web Uygulaması

Senaryo1'de belirtilen, kullanıcıların iş isteklerini web arayüzleri aracılığı ile sisteme iletebilmesi için Şekil 3.1 deki gibi birçok web uygulama tasarımında da kullanılan katmanlı bir yapı kullanılmıştır. En alt katmanda sistemdeki verileri tutacak veritabanı sistemi bulunmaktadır. Veritabanı olarak MYSQL kullanılmasına karar verilmişti. Bir üst katmanda veritabanı sistemindeki verilere ulaşabilmek için veri erişim katmanı oluşturulmuş ve burada da Hibernate teknolojisi kullanılmıştır. İş katmanında ise tüm servis arayüzleri ve iş akışlarının bulunduğu tüm algoritmaları içeren bölümler bulunmaktadır. Burada J2EE bütünün içerisinde yer alan Java nesneleri kullanılmıştır. Sunum katmanı ise kullanıcı ile etkileşim içerisinde olan ve kullanıcıdan aldığı bilgileri belirli kontrolleri yaptıktan sonra iş katmanına iletmekle sorumlu birimdir. Burada Struts ve JSP teknolojisinden yararlanılmıştır. Bütün

katmanlarda kullanılabilen ve sistem içerisinde yapılan tüm iş adımlarını loglamak için log4J teknolojisi kullanılmıştır.

4.2 Arka Planda Çalışan Uygulamalar

Arka planda bir zamanlayıcı ile birlikte çalışacak 3 farklı uygulama bulunmaktadır. Bu uygulamaların üçü de yukarıdaki altyapının sadece sunum katmanı hariç, diğer tüm katmanları üzerinde çalışmaktadır. Senaryo2 ve Senaryo3'deki süreçlerin gerçekleştirilmesi bu 3 uygulama parçacığı tarafından yapılmaktadır.

4.2.1 Grid ftp yönetimi

Senaryo3'de belirtilen, çalıştırılacak kaynak kodların ve giriş verilerinin kaynak sisteme ulaştırılması gerekmektedir. Bu dosyaların uzak bir makineye gönderilmesi için FTP protokolünün uygun olduğu görülmüş ve tasarım aşamasında bunun kullanılması karar verilmiştir. Bu uygulama parçacığı Quartz zamanlayıcı teknolojisini kullanarak belirli zaman aralıkları ile daha önce rezervasyonu yapılmış ve henüz çalışmaya başlamamış işleri kendisine alır ve belirli bir plan çerçevesinde işin gönderilip gönderilmeyeceğine karar verir. Gönderilmesi gereken işlerin bilgilerini veritabanından okuyarak dosya sistemi üzerinden iş için gerekli dosyaları alarak kaynak sisteme FTP protokolü ile bağlanarak dosyaları bırakır. Kaynak daha sonra iş için bildirim geldiğinde kendisinde bulunan bu dosyaları alarak işleme başlar. Bu bölümdeki uygulama parçacığı Şekil 3.1'de belirtilen yazılım katmanlarından sunum katmanı hariç diğer tüm katmanlar üzerinde çalışır.

4.2.2 Grid iş gönderme yönetimi

Senaryo2'de belirtilen, kullanıcılardan işlerini çalıştırmak için ileri tarihli alınan rezervasyonları, yine Quartz teknolojisi ile belirli zaman aralıkları ile kontrol ederek zamanı gelen işlerin listesi çekilir. Bu liste içerisindeki işleri paralel olarak kaynaklara göndermeye başlar. Burada Axis2 web servis teknoloji kullanarak kaynak tarafından daha önceden verilmiş olan web servis arayüzünü kullanarak iş ile ilgili ayrıntıları kaynak sisteme iletir. Aynı şekilde bu isteği alan kaynak sistem, işin belirli aşamalarında kaynak yönetim sistemi tarafından daha önce kendisine verilmiş web servis arayüzünü kullanarak işin kendisine ulaştığına, işe başladığına ve işin bittiğine dair güncellemelerde bulunur. Kaynak sistem üzerindeki işi başarılı bir şekilde

bitirdiğinde oluşan çıktı dosyalarını yine FTP protokolü aracılığı ile kullanıcı tarafından belirtilen kaynağa gönderilir.

4.2.3 Grid bilgi servis yönetimi

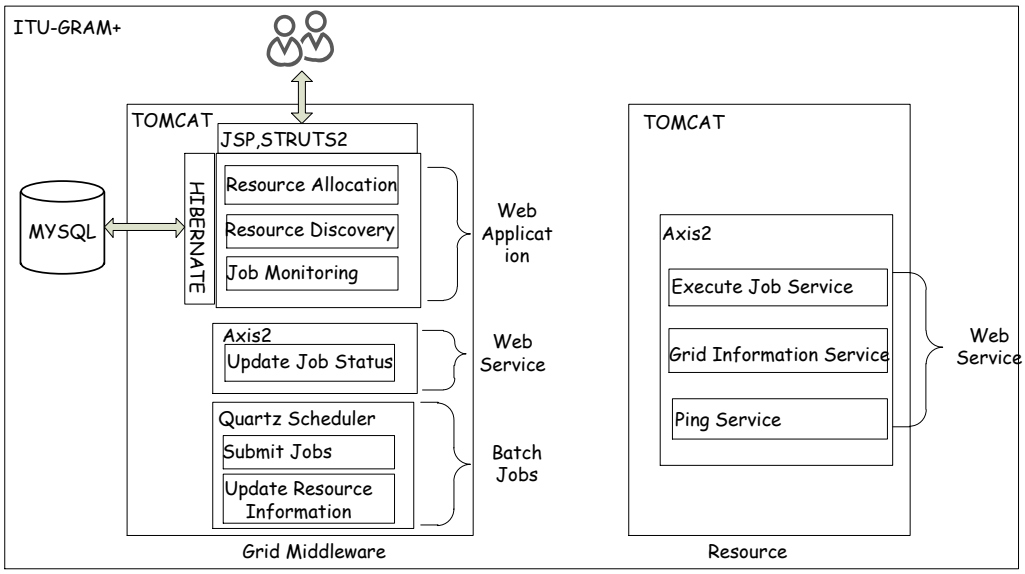
Kaynak sistem üzerindeki kaynakların dinamik olarak değişebileceğini ve bu bilgilerin doğru seçimler yapılabilmesi için sürekli güncel olarak tutulması gerektiği daha önceki bölümlerde belirtilmişti. Bu uygulama parçacığı kaynakların üzerinde çalışan bir web servis uygulamasına erişerek, kaynak sistem üzerindeki kaynakların durumunu belirli zaman aralıkları ile sorgular. Sorgu sonucunda elde edilen bilgiler veritabanı üzerinde güncellenmektedir. Buradaki uygulama parçacığı yukarıdaki Şekil 3.1'de belirtilen sunum katmanı hariç diğer katmanlar üzerinde çalışır.

4.3 Grid Kaynak Ayırma Yönetimi

Kullanıcılardan iş ile ilgili ayrıntılar alındıktan sonra kullanıcının statik olarak vermiş olduğu kaynak bilgilerine göre bir eleme yapılır ve uygun kaynak listesi oluşturulur, bu liste üzerinden kullanıcının vermiş olduğu işin son teslim tarihi bilgisine göre bugünden o tarih aralığına kadar liste içerisinde daha önce rezerve edilmiş olanlar elenir ve kalan liste içerisinde belirli kurallara göre her bir kaynak için finansal bir değer hesaplanır ve tüm liste kullanıcı ile paylaşılır. Kullanıcı bütçesine uygun ve belirtilen süre içerisinde işlemi yapabilecek olan kaynağı seçer. Artık o kaynak o tarihler arasında o kullanıcıya rezerve edilmiştir.

5. ITU-GRAM+ GERÇEKLENMESİ

Bu bölümde tasarımı yapılan kaynak yönetim sisteminin gerçeklenmesi anlatılacaktır. Şekil 5.1'de ITU-GRAM+ içerisinde yer alan tüm bileşenler ve bu bileşenler için yazılmış web uygulaması ve web servisler ayrıntıları ile verilmiştir.



Şekil 5.1 : ITU-GRAM+ mimarisi.

İlk bölümde, kullanılan veritabanı sistemi ve ilişkileri anlatılmıştır. İkinci bölümde gerçekleştirilen web uygulaması, web servisler ve arka planda çalışan uygulamalar anlatılmıştır. Son bölümde ise iş ve kaynak ile ilgili durum diyagramları verilmiştir.

5.1 Veritabanı Sistemi ve İlişkileri

Kaynak yönetim sistemlerinin, kaynak ile ilgili statik ve dinamik bilgileri, kullanıcıdan alınmış olan iş ile ilgili talepleri ve kullanıcı bilgilerini tutabileceği veritabanına ihtiyaç vardır. ITU-GRAM+'da kullanılan teknolojiler kısmında da açıklandığı gibi veritabanı olarak Mysql seçmiştir. Tasarıma göre sistemde olması gereken nesneler oluşturulmuş ve bu nesnelerin özellikleri ile birlikte bir veritabanı sistemi yaratılmıştır. Veritabanı şeması ve nesnelerin birbirileri ile olan ilişkileri Şekil 5.2'de gösterilmiştir. Beş temel nesne bulunmaktadır.

Kullanıcı(User): Kaynak yönetim sisteminden faydalanacak olan kişilerdir. Kullanıcıların sisteme giriş yapabilmeleri için gerekli bilgileri tutar.

Kaynak(Resource): Kaynağın *ip*, *hostname* gibi statik bilgilerinin yanında dinamik olarak değişebilen *cpu*, *storage* bilgilerini de içerir.

Ayrıca her kaynak değişen bu dinamik bilgilere göre ana bir maddi değere sahiptir.

İş(Job): Kullanıcının JDL dosyası ve kullanıcı arayüzü ile girmiş olduğu ITU-GRAM+'a göndereceği iş ile ilgili tanımlamaları içerir.

Rezervasyon(ResReservationTimeTable): Hangi işe, hangi kaynağın, hangi süreler içerisinde ayrıldığı bilgisini içerir. Bu tablo sistemde en çok değişikliğe uğrayan tablodur. Süreç içerisinde yer alan tüm adımlarda buradaki durum bilgisi güncellenir ve bu durum bilgisine göre yeni işlemler alınır.

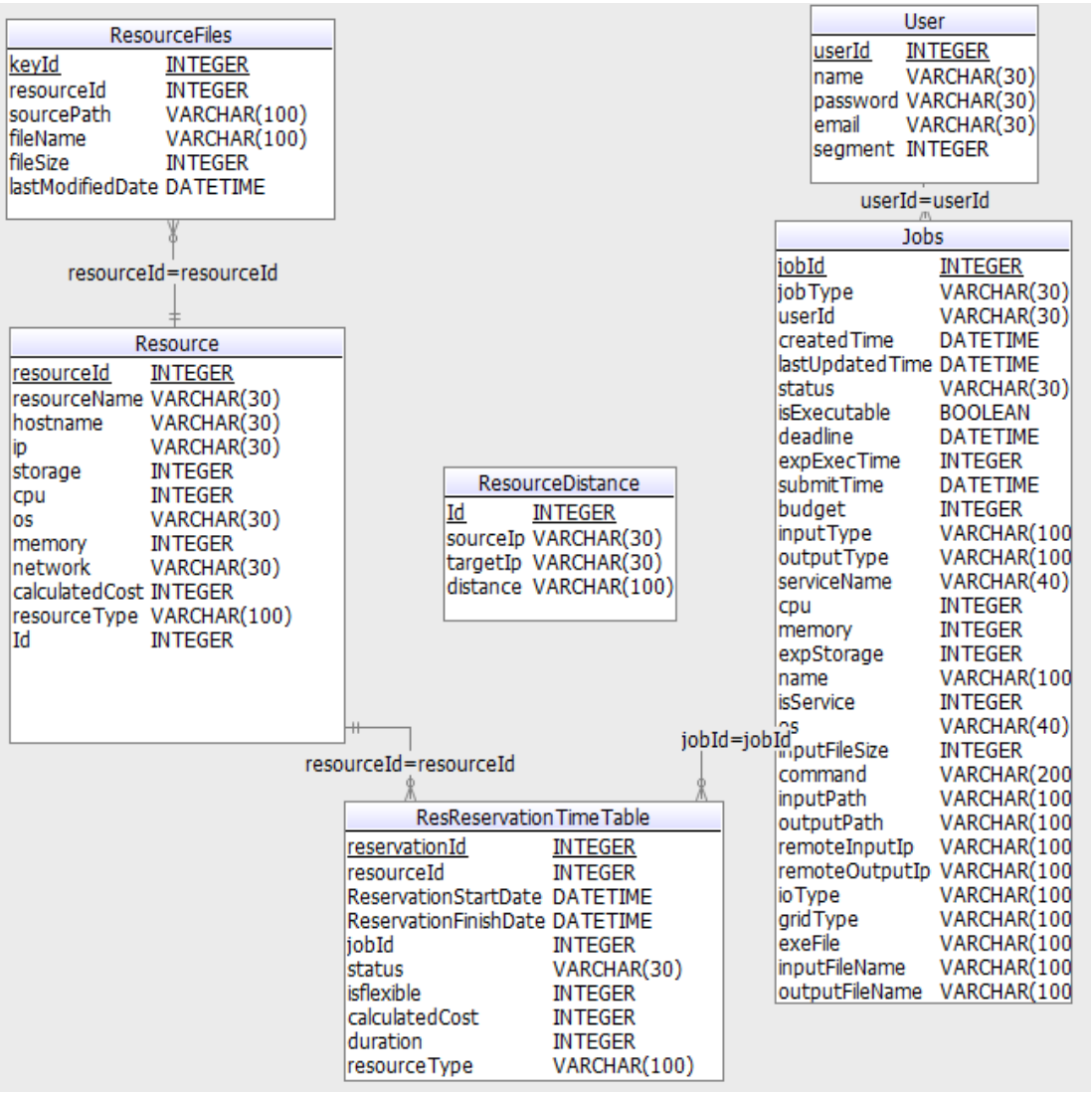
Kaynak Dosyaları(ResourceFiles): Kaynak üzerinde bulunan dosyalar ile ilgili bilgileri içerir.

KaynakUzaklık(ResourceDistance): İki kaynak arasındaki uzaklık bilgisini geçmişe yönelik tutmak için kullanılır.

```
<hibernate-mapping>
<classname="dal.ResourceAssoc"table="ResReservationTimeTable">
<idname="reservationId"type="int">
<columnname="reservationId"/>
<generatorclass="increment"/>
</id>
<propertyname="resourceId"type="int">
<columnname="resourceId"length="0"/>
</property>
<propertyname="ReservationStartDate"type="timestamp">
<columnname="ReservationStartDate"length="0"/>
</property>
<propertyname="ReservationFinishDate"type="timestamp">
<columnname="ReservationFinishDate"length="0"/>
</property>
<propertyname="assignedJobId"type="int">
<columnname="assignedJobId"length="0"/>
</property>
<propertyname="isflexible"type="int">
<columnname="isflexible"length="0"/>
</property>
<propertyname="status"type="string">
<columnname="status"length="64"/>
</property>
<propertyname="calculatedCost"type="int">
<columnname="calculatedCost"length="0"/>
</property>
<propertyname="duration"type="int">
<columnname="duration"length="0"/>
</property>
</class>
</hibernate-mapping>
```

Şekil 5.2 : Örnek Hibernate XML dosyası.

Veritabanı nesneleri ve birbirleri ile olan ilişkileri tanımlandıktan sonra bu nesnelere ait POJO sınıfıları (Plain Old Java Objects) yaratılmıştır ve veritabanı nesneleri ile Java sınıfları arasında ilişki kuran Hibernate XML dosyaları oluşturulmuştur. Örnekleme amacı ile *Rezervasyon* nesnesine ait Hibernate XML dosyası Şekil 5.2'de gösterilmiştir. İlk satırda hangi Java sınıfına ait olduğu belirtilir. Daha sonra her bir veritabanı kolonuna karşılık gelen Java sınıfındaki alanlar birbirleri ile eşleştirilir. Böylece bundan sonra her veritabanı işlemi sırasında doğrudan olarak veritabanına ulaşıp işlem yapılmadan eşleştirme yapılan bu Java sınıfları ile işlem yapılır. Bu özellikle ileride değişebilecek kolonlar üzerinde değişiklik yapıldığında sadece bu sınıflar üzerinde değişiklik yapılarak hızlı ve en az etkilenecek şekilde bir yapı oluşturulur.



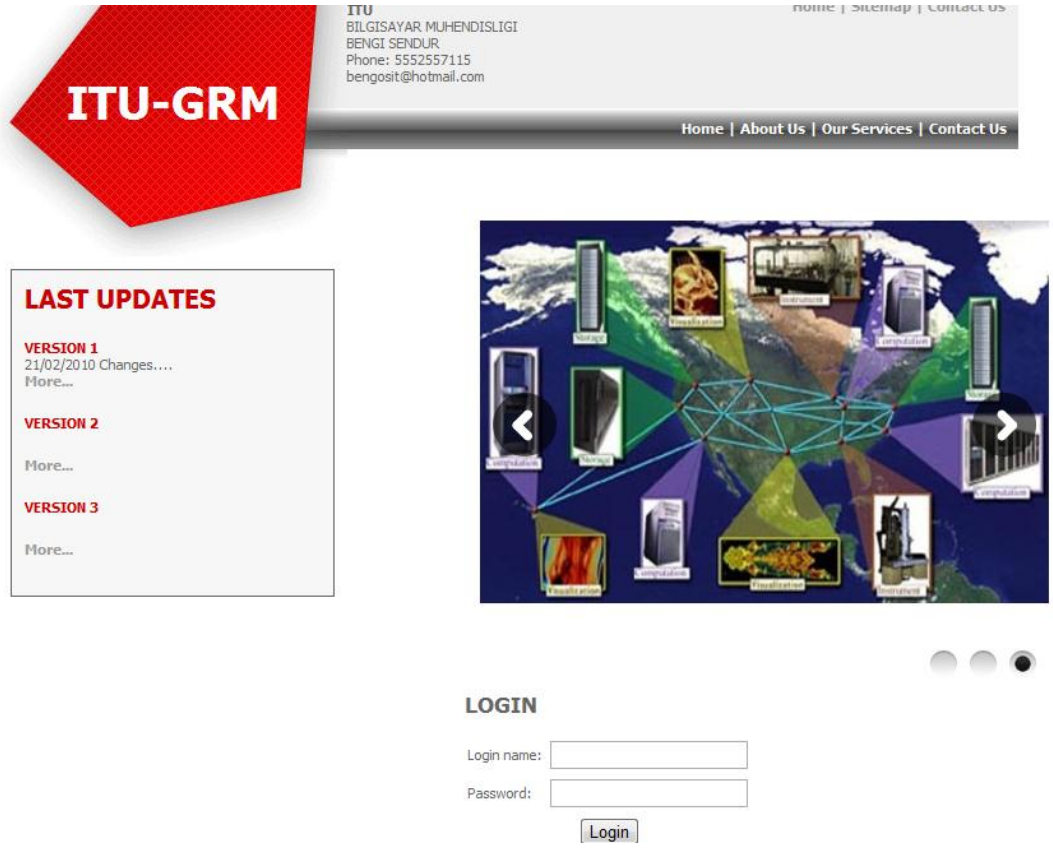
Şekil 5.3 : Veritabanı sistemi ve ilişkileri.

5.2 Grid Web Uygulaması

Bu bölümde tasarım aşamasında ortaya çıkan senaryolar içerisinde kullanıcıların sisteme giriş yapması ve iş taleplerini iletmesi, kaynakların ilgili iş taleplerine göre rezerve edilmesinin nasıl gerçeklendiği anlatılmıştır.

5.2.1 Sisteme kayıt ve giriş

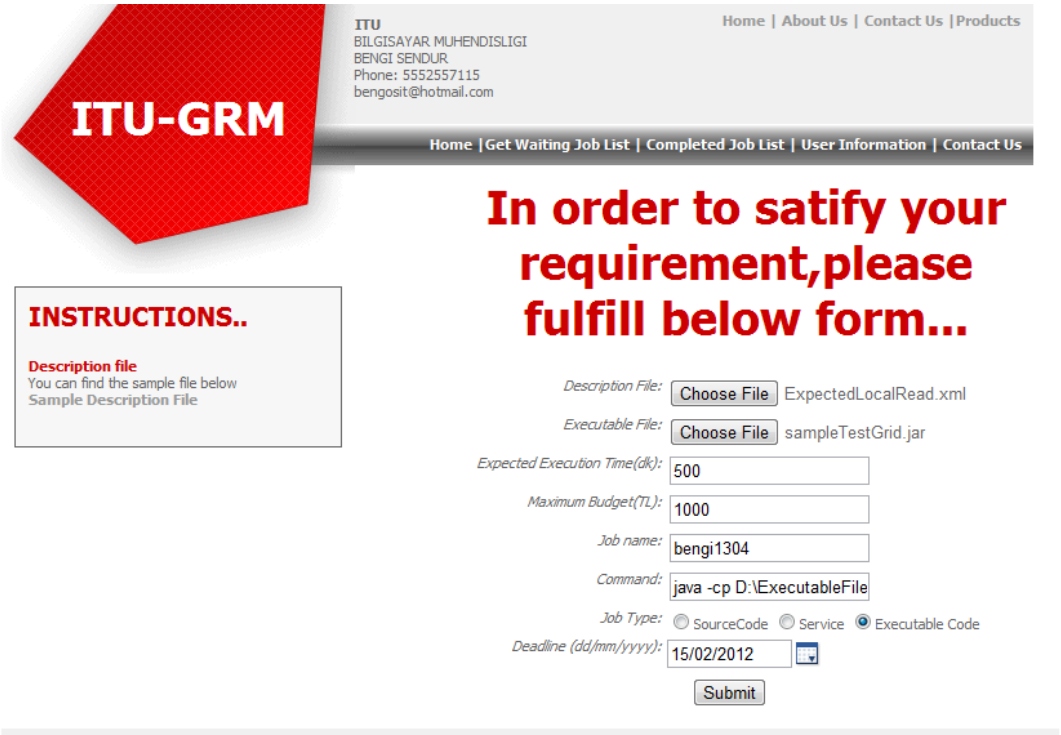
Kullanıcıların sisteme kayıt ve giriş yapmaları için web arayüzleri tasarlanmıştır. Kullanıcılar sisteme kayıt sırasında oluşturdukları kullanıcı adı ve şifreleri ile sisteme giriş yaparlar. Kullanıcı bir defa sisteme doğru bir şekilde giriş yaptıktan sonra kullanıcı kimliği, oturum bilgisi içerisine alınır ve daha sonra yapacağı işlemler otomatik olarak kullanıcı kimliği ile ilişkilendirilir. Böylece kullanıcın web sayfaları üzerinde yaptığı her türlü işlem sisteme aktarılır. Sistem bu aşamadan sonra yapılacak her işlemde bu oturum kimliğini kontrol eder ve bu şekilde izinsiz girişleri önlemiş olur. Şekil 5.4'de ITU-GRAM+ üzerinden alınan giriş sayfası ekran görüntüsü bulunmaktadır.



Şekil 5.4 : ITU-GRAM+ giriş sayfası.

5.2.2 Kullanıcı iş taleplerini alma

Kullanıcıdan bazı parametreleri web arayüzünden girmelerini bazı parametreleri de daha önceden tanımlanmış olan JDL(Job Description Language) dosyası içerisinde girmesi istenir. XML dosyası örneği şekil 5.6'da verilmiştir. Kullanıcıdan bu basit XML dosya örneğini, kaynak kod veya çalıştırılabilir kod parçasını, uygulamanın ne kadar süre çalışacağı, en son teslim tarih bilgisi ve ekranda görülen diğer bilgiler alındıktan sonra sistem tarafından yapılan işlemlerin sırası adım adım gösterilmiştir. Şekil 5.5'de iş taleplerinin alındığı örnek bir ekran verilmiştir.



Şekil 5.5 : İş taleplerini alma ekranı.

Şekil 5.6'da XML yapısındaki dosya içerisinde iş ile ilgili tanımlamalar yapılabilmektedir. Her bir alan ile ilgili olarak bir üst kısmında yorum satırları bulunmaktadır. Böylece kullanıcılar iş taleplerine göre hangi alanları doldurup doldurmayacağını bu bilgilere bakarak kolaylıkla yapabilir. Bu yapıdaki bir dosya hem kullanıcılar açısından kolaylık oluşturmakta hem de uygulamanın kullanılabilirliğini arttırmaktadır. Aşağıda alanların açıklamaları ve nasıl kullanılacakları ile ilgili bilgiler ayrıntılı olarak verilmiştir.

- gridType: Sistemin hangi kaynağa daha çok kullanılacağının bilgisini içermektedir. Örneğin işlemin giriş/çıkış dosyaları küçük ölçekli olmasına

rağmen yapılacak işlem içeriği çok fazla CPU gerektiriyor ise bu durumda grid tipi *Computing Intensive* yani işlem yoğunluklu olarak algılanmaktadır. Fakat tam tersi olarak yapılacak işlem içeriği düşük fakat kullanılacak giriş dokümanları ile çıkış dokümanlarının içeriği yüksek ise *Data Intensive* grid tipi olarak belirtilmelidir. Tüm grid işlemler bu şekilde keskin hatlar ile ayrılamadığı için *Both* diyerek aslında datanın ve işlemin büyüklüğünün birbirine göre çok yakın olduğu belirtilebilir.

```
<?xml version="1.0" encoding="UTF-8"?>
<RequirementList>
    <gridType>DataIntensive</gridType>
    <ioType>Read</ioType>
    <inputFileType>Expected File</inputFileType>
    <inputFileName>deprem.txt</inputFileName>
    <inputfilePath>D:/Veriler/Test</inputfilePath>
    <remoteMachine>192.168.1.1</remoteMachine>
    <isWrited>No</isWrited>
    <os>windows</os>
    <cpu>1</cpu>
    <memory>3</memory>
    <storage>200</storage>
    <outputFileType>Local</outputFileType>
    <outputFileName>output.txt</outputFileName>
    <outputfilePath>D:/Veriler/Test</outputfilePath>
    <remoteOutputMachine>192.168.1.1</remoteOutputMachine>
</RequirementList>
```

Şekil 5.6 : JDL dosya örneği.

- ioType: Yapılacak i/o işlemlerinin daha çok okuma/yazma ağırlık olarak yapıldığı ile ilgilidir. Örneğin işleme girecek olan giriş dosyası çıkış dosyasına oranla daha büyük ise seçilecek kaynak giriş dosyasına yakın olacak yerde seçilir ki ağ üzerinde geçen süre maliyetlerinden kazanım sağlansın.
- inputFileType: Grid uygulaması için gerekli olan giriş dosyasının nereden bulunacağı veya nereden alınacağı bilgisini içerir. Bu etiket işleme girecek giriş dosyasının yerel kaynakta mı, belirtilen başka bir kaynakta mı yada Grid sisteminin içerisinde ve sistemden bu dosyanın bulunmasının istenmesine göre üç farklı değer alabilir. Burada girilen bilgilere göre kullanılan algoritma değişiklik göstermektedir.

- inputFileName: Tüm giriş tipleri için bu değerin girilmesi beklenmektedir. Dosya ismi giriş tipine göre ya belirtilen makine içerisinde alınır ya da Grid sistemi içerisindeki dosyalardan bulunması isteniyor.
- inputfilePath: Seçilen giriş tipine göre zorunludur. Giriş tipi olarak *Local* veya *Remote* seçildiğinde kullanıcının dosyanın nerede olduğunu belirtmesi gerekir.
- remoteMachine: Giriş tipi olarak *Local* veya *Remote* seçildiğinde dosyanın hangi makineden okunacağı bilgisini verir.
- os: Çalıştırılacak işlem için özel bir işletim sistemi gerekli ise bu alanın doldurulması zorunludur.
- cpu: Yapılacak grid işleminin ne kadar cpu gerektiği bilgisini içerir. Doldurulması zorunludur.
- memory: Yapılacak grid işleminin ne kadar belleğe ihtiyacı olduğu bilgisini içerir.
- storage: Yapılacak grid işleminin ne kadar büyüklükte bir depolama ihtiyacı olduğu bilgisini içerir.
- outputFileType: Grid işlemi sonucu oluşacak dosyanın nerede saklanacağı ile ilgili bilgi verir. Örneğin oluşan dosya yerel makinede veya bilinen bir makine üzerinde saklanılması istenirse saklamak için herhangi bir kaynak ataması yapılmayacaktır.
- outputFileName: Grid işlemi sonucu oluşacak dosyanın isim bilgisini içerir. Örneğin oluşturacak dosya grid işlem sonucunda verilen bu isim bilgisi ile karşılaştırılacak ve eğer eşlenirse belirtilen çıkış tipine göre ilgili yere gönderilecektir.
- outputfilePath: Grid işlemi sonucu oluşacak dosyanın veri yolunu belirtir.
- remoteOutputMachine: Grid işlemi sonucu oluşacak dosyanın hangi makine üzerinde tutulacağı bilgisini içerir.

  Uygulama içerisinde kullanıcıların erişebileceği bir link ile bu dosyanın örnek bir hali sisteme konulmuştur. Kullanıcı buradaki yorum satırlarına bakarak kendi işi ile ilgili bilgileri doldurur.

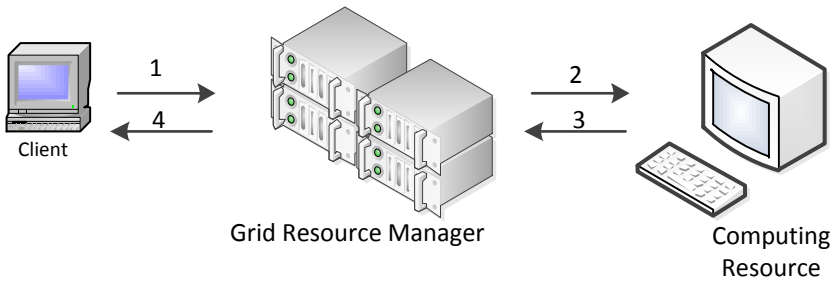
5.2.3 Uygun kaynakları bulma

İş taleplerine göre uygun kaynak bulma algoritması çeşitlilik göstermektedir. Aşağıda oluşabilecek farklı durumlar için nasıl bir algoritma oluşturulduğu ve ayrıntıları incelenecektir. Bir önceki bölümde verilen xml içerisinde grid tipi bilgisine göre algoritmaya başlanır. .Veri yoğunluklu işlemde belirtilmek istenen giriş veya çıkış dosyalarının büyük olduğu ve bu sebeple de işlem yoğunluklu gridler ile karşılaştırıldığında kaynak bulmanın farklı parametrelere dayandığı söylenilmelidir.

.Bir sonraki bölümde, dosyaların giriş ve çıkış tiplerine göre oluşabilecek tüm durumlar incelenmiştir.

5.2.3.1 Giriş ve çıkış dosyalarının yerel kaynakta tutulması

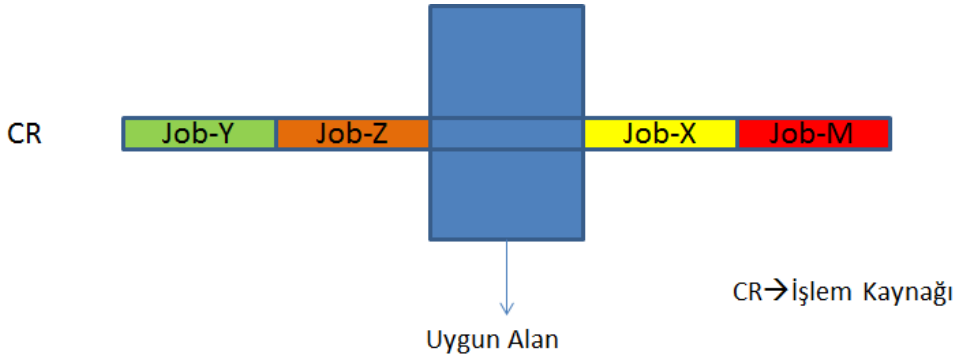
Giriş ve çıkış dosyalarının yerel kaynakta tutulması grid içerisinde kaynaklardan sadece işlem yapılmak için CR(İşlem Kaynağı) kullanılacağı anlamına gelmektedir. Şekil 5.7'de işlem sırasının nasıl olduğu gösterilmiştir. Kullanıcı tarafından iş bilgileri kaynak yönetim sistemine gönderilir. Sonrasında sadece CR kaynağı ihtiyaç olacağı için iş ona gönderilir ve sonuç tekrardan kullanıcı ile paylaşılır.



Şekil 5.7 : İşlem kaynağı grid döngüsü.

Bu durumda sadece yerel kaynaklar ile çalışıldığı için işin çalıştığı sürece boyunca belirtilen kaynakların ulaşılabilir olması beklenmektedir. Aksi halde işlem başarısız olacaktır.

Şekil 5.8'de örnek olarak sadece bir CR gösterilmiştir. Grid kaynak yönetim sistemi bunun gibi uygun olan bütün CR kaynaklarını bulur .Fakat teknik olarak uygun olan tüm kaynakları listelemek yerine maliyet açısından kullanıcının bütçesine uyan en uygun kaynak listesinden en fazla 10 tanesini ekrana listeler ve kullanıcdan bunların arasındann seçim yapması istenir.



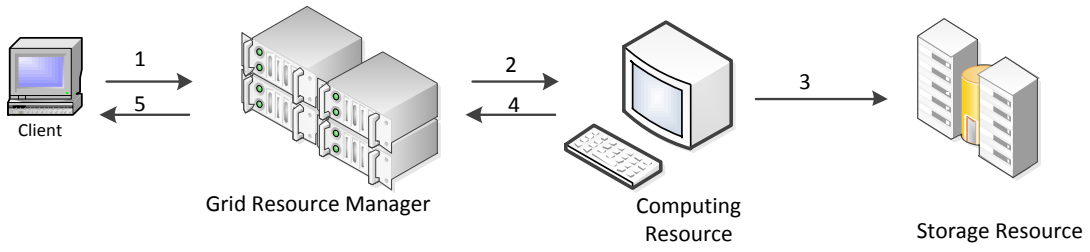
Şekil 5.8 : İşlem kaynağı -uygun alan.

Pseudocode

```
CR Listesi ← FindEligibleCR(Job)
While(CR in CR List)
Begin
CR Reserv List←CheckRes(CR)
     CR Available List←FindEmptySlot(CR Reserv List,duration)
      If(CR Available List is not null)
           genList←AddGeneralList(CR Available List)
End
Display(genList)
```

5.2.3.2 Giriş dosyasının yerel ve çıkış dosyasının grid sistemde tutulması

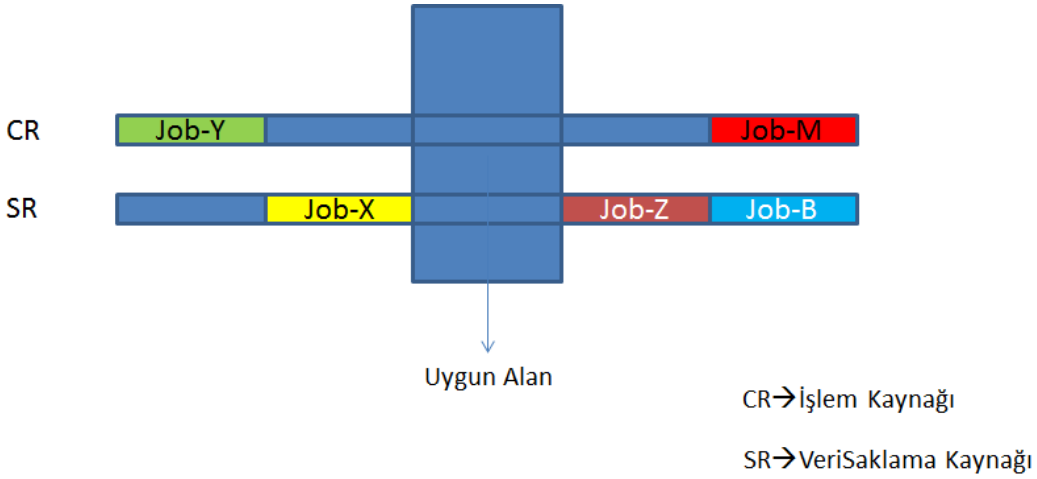
Giriş dosyasının yerel sistemden alınması ve çıkış dosyasının grid içerisinde kaynaklarda saklanması, işlemin yapılması için CR(İşlem Kaynağı) ve verinin saklanması için ve SR'a (Veri Depolama Kaynağı) ihtiyaç duyulur.



Şekil 5.9 : Depolama kaynağı grid döngüsü.

Şekil 5.9'da işlem sırasının nasıl olduğu gösterilmiştir. Dosyalar yerel kaynaktan alınacak ise rezervasyon süresi boyunca yerel kaynağın ulaşılabilir olması

gerekmektedir. Aksi takdirde işlem başarısız olacaktır. Burada yapılacak işlemin i/o olarak ne kadar okuma/yazma yapacağı önemlidir. Eğer yazma işlemi okuma işlemine göre daha yoğunluklu ise bir başka deyişle giriş dosyasının boyutu çıkış dosyasının boyutuna göre daha küçük ise ağ üzerinde geçen süreyi azaltmak amacıyla, seçilecek CR, yerel makineden daha çok SR'a yakın olacak şekilde seçilir. Şekil 5.10'da örnek olarak sadece bir CR ve SR eşleştirilmesi gösterilmiştir.



Şekil 5.10 : İşlem ve depolama kaynağı-uygun alan.

Grid kaynak yönetim sistemi bunun gibi uygun olan bütün kaynak çiftlerini bulur ve maliyet olarak kullanıcının bütçesiyle uyan uygun kaynakların listesini gösterir.

Pseudocode

```
CR List ← FindEligibleCR(Job)
SR List ← FindEligibleSR(Job)
While(CR in CR List)
Begin
      CR Reserv List ← CheckRes(CR)
     CR Available List←FindEmptySlot(CR Reserv List,duration)
      İf(CR Available List is not null)
             While(SR in SR List)
                    Begin
                   SR Reserv List ← CheckRes(SR)
                    SR Available List ← FindEmptySlot(SR Reserv
                                  List,CR Reserv List,duration)
                       İf(SR Available List is not null)

 genList←AddGeneralList(CR Available List, SR Available List)
```

```
      End
CalculateCost(genList)
Display(genList)
```

5.2.3.3 Giriş dosyasının yerel ve çıkış dosyasının belirli sistemde tutulması

Giriş dosyasının yerel sistemden alınması ve çıkış dosyasının kullanıcının belirtmiş olduğu ilgili kaynak ve dosya içerisinde tutulması, işlemin yapılması için CR(İşlem Kaynağı) ve verinin saklanması için SR(Veri Depolama Kaynağına) ihtiyaç duyulduğu anlamına gelmektedir. Fakat bir öncekinden farklı olarak dosyanın saklanacağı sistem bilgisi kullanıcı tarafından belirtilir. Bu durumda Grid sistem belirtilen kaynağın Grid sistem içerisinde yer alıp almadığını kontrol eder. Eğer sistemde yer alıyor ise onu kaynak olarak rezerve eder. Eğer grid sistem içerisinde yer almıyor ise işlem süresi boyunca belirtilen makinenin ulaşılabilir olması gerekmektedir. Aksi takdirde işlem başarısız olacaktır. Burada yapılacak işlemin i/o olarak ne kadar okuma/yazma yapacağı önemlidir. Eğer yazma işlemi okuma işlemine göre daha yoğunluklu olacak ise bir başka söyleyiş ile, giriş dosyasının boyutu çıkış dosyasının boyutuna göre daha küçük olacak ise ağ üzerinde geçen süreyi azaltmak amacıyla seçilecek CR, yerel makineden daha çok SR'a yakın olacak şekilde seçilir. Çıktı için belirtilen kaynak grid içerisinde ise grid döngüsü 5.9'daki gibi olacaktır. Çıktı için belirtilen kaynak grid içerisinde değil ise grid döngüsü 5.7'deki gibi olacaktır. Şekil 5.10'da örnek olarak sadece bir CR ve SR eşleştirilmesi gösterilmiştir. Grid Kaynak sistemi bunun gibi uygun olan bütün kaynakları bulur ve maliyet olarak kullanıcının bütçesiyle uyan uygun kaynakların listesini ekrana basar. Aşağıda bununla ilgili olan kod parçası verilmiştir.

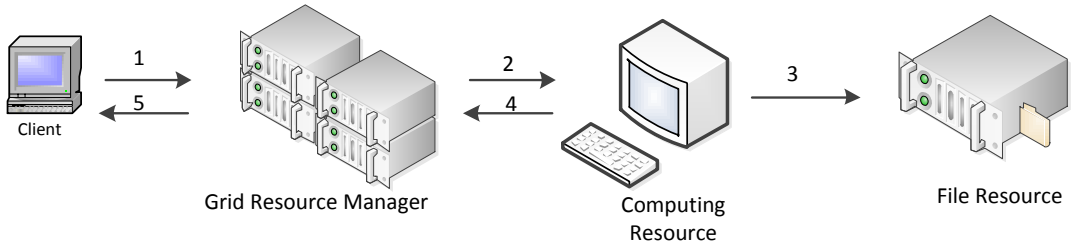
Pseudocode

```
CR List ← FindEligibleCR(Job)
İf(SR is in Grid System)
    SR List ← FindEligibleSR(Job)
While(CR in CR List)
     Begin
     CR Reserv List ← CheckRes(CR)
     CR Available List←FindEmptySlot(CR Reserv List,duration)
     İf(CR Available List is not null)
          İf(SR is in Grid System)
```

```
      While(SR in SR List)
      Begin
SR Reserv List ← CheckRes(SR)
                    SR Available List ← FindEmptySlot(SR Reserv
                                        List,CR Reserv List,duration)
                                  İf(SR Available List is not null)
                                genList←AddGeneralList(CR Available
                                          List, SR Available List)
       End
 Else
            genList←AddGeneralList(CR Available List)
CalculateCost(genList)
Display(genList)
```

5.2.3.4 Giriş dosyasının belirli kaynaktan alınması ve çıkış dosyasının yerel kaynakta tutulması

Giriş dosyasının kullanıcı tarafından belirtilen kaynaktan alınması ve çıkış dosyasının yerel kaynakta saklanması, işlemin yapılması için CR(İşlem Kaynağı) ve girdi dosyasının alınması için FR'a (Dosya Kaynağı) ihtiyaç duyulduğu anlamına gelmektedir. Şekil 5.11'de bununla ilgili olarak işlem sırası gösterilmiştir.

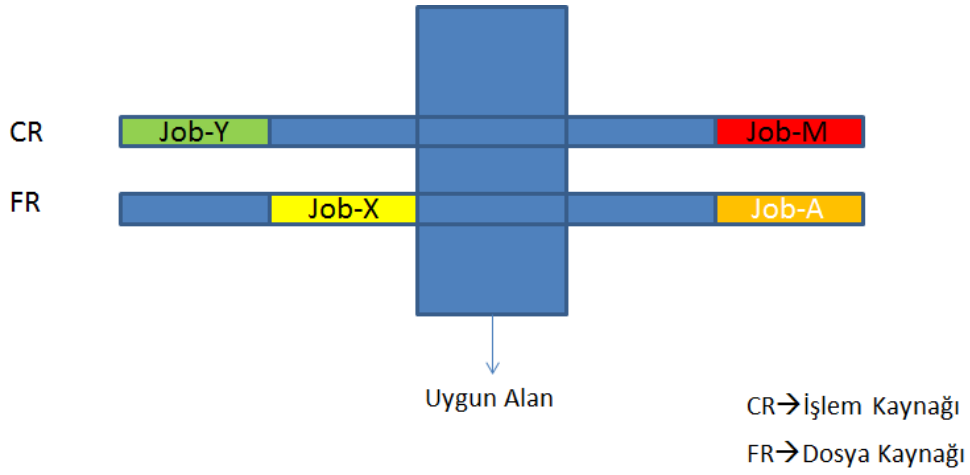


Şekil 5.11 : İşlem ve dosya kaynağı döngüsü.

Dosyanın bulunacağı kaynak bilgisi kullanıcı tarafından verilir. Bu durumda sistem belirtilen kaynağın Grid sistem içerisinde yer alıp almadığını kontrol eder. Eğer sistemde yer alıyor ise onu kaynak olarak rezerve eder eğer grid sistem içerisinde yer almıyorsa rezervasyon işlemi yapılamaz fakat işlem süresi boyunca kaynağın ulaşılabilir olması beklenir. Aksi takdirde işlem başarısız olacaktır. Burada yapılacak işlemin i/o olarak ne kadar okuma/yazma yapacağı önemlidir. Eğer girdi dosyasının

boyutu çıktı dosyasına oranla daha büyük ise bu durumda seçilecek CR ağ üzerinde geçen süreyi azaltmak adına FR'a yakın olacak şekilde seçilecektir.

Girdi için belirtilen kaynak grid içerisinde ise;



Şekil 5.12 : İşlem ve dosya kaynağı-uygun alan.

Girdi için belirtilen kaynak grid içerisinde değil ise uygun alan şekli 5.6'daki gibi olacaktır.

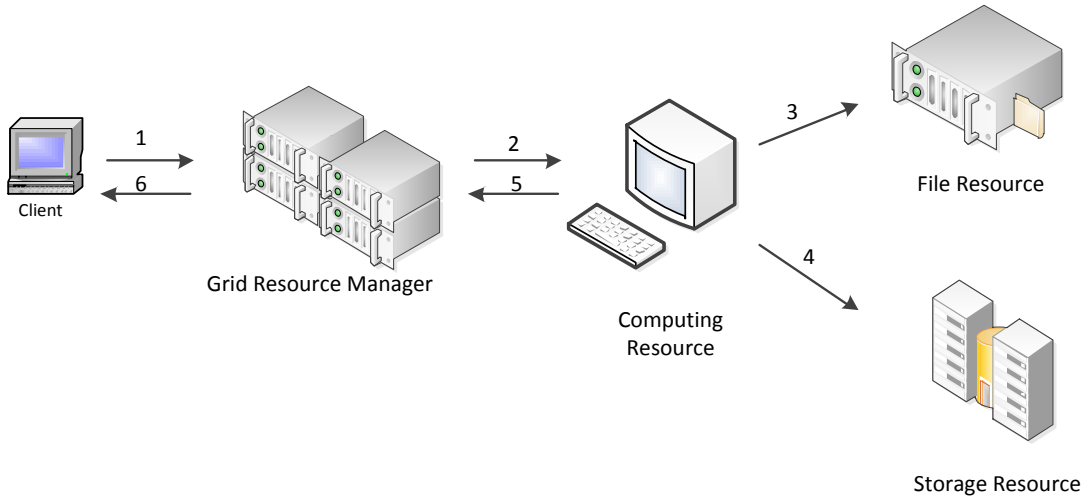
Şekil 5.12'de örnek olarak sadece bir CR ve FR eşleştirilmesi gösterilmiştir. Grid Kaynak sistemi bunun gibi uygun olan bütün kaynakları bulur ve maliyet olarak kullanıcının bütçesiyle eşleşen uygun kaynakların listesini gösterir. Aşağıda bununla ilgili olan kod parçası verilmiştir.

```
CR List ← FindEligibleCR(Job)
İf(FR is in Grid System)
    FR List ← FindEligibleFR(Job)
While(CR in CR List)
Begin
     CR Reserv List ← CheckRes(CR)
     CR Available List←FindEmptySlot(CR Reserv List,duration)
     İf(CR Available List is not null)
          İf(SR is in Grid System)
     While(SR in SR List)
     Begin
FR Reserv List ← CheckRes(FR)
```

```
                    FR Available List ← FindEmptySlot(FR Reserv
                                        List,CR Reserv List,duration)
                              İf(FR Available List is not null)
                                genList←AddGeneralList(CR Available
                                                    List, FR Available List)
        End
          Else
               genList←AddGeneralList(CR Available List)
CalculateCost(genList)
Display(genList)
```

5.2.3.5 Giriş dosyasının belirli kaynaktan alınması ve çıkış dosyasının belirtilen kaynakta tutulması

Giriş dosyasının kullanıcı tarafından belirtilen kaynaktan alınması ve çıkış dosyasının kullanıcı tarafından belirtilen kaynakta saklanması, işlemin yapılması için CR(İşlem Kaynağı) ve girdi dosyasının alınması için FR(Dosya Kaynağına) ve çıktı dosyasının saklanması için de SR'a(Veri Depolama Kaynağına) ihtiyaç duyulduğu anlamına gelmektedir. Şekil 5.13'de bununla ilgili olarak işlem sırasının nasıl olacağı gösterilmiştir.

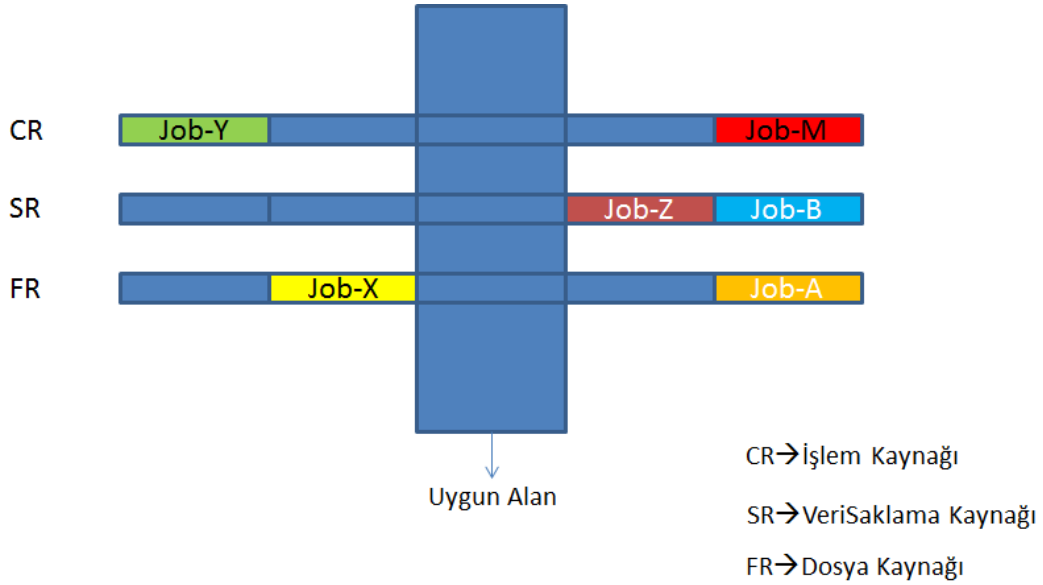


Şekil 5.13 : İşlem, depolama ve dosya kaynağı grid döngüsü.

Bu durumda sistem belirtilen kaynakların Grid sistem içerisinde yer alıp almadığını kontrol eder. Eğer sistemde yer alıyor ise onu kaynak olarak rezerve eder eğer grid sistem içerisinde yer almıyorsa rezervasyon işlemi yapılamaz. İşlem süresi boyunca kaynakların ulaşılabilir olması beklenir .Aksi takdirde işlem başarısız olacaktır.

Burada yapılacak işlemin i/o olarak nekadar okuma/yazma yapacağı önemlidir. Eğer girdi dosyasının boyutu çıktı dosyasına oranla daha büyük ise bu durumda seçilecek CR ağ üzerinde geçen süreyi azaltmak adına FR'a yakın olacak şekilde seçilecektir. Tam tersi durumda ise CR, FR'a yakın olacak şekilde seçilir.

Girdi ve çıktı için belirtilen kaynaklar grid içerisinde ise uygun alan Şekil 5.14'deki gibi olacaktır.



Şekil 5.14 : İşlem, depolama ve dosya kaynağı-uygun alan.

Sadece girdi için belirtilen kaynak, grid içerisinde değil ise grid döngüsü Şekil 5.10'daki gibi olacaktır.

Sadece çıktı için belirtilen kaynak grid içerisinde değil ise grid döngüsü Şekil 5.12'deki gibi olacaktır.

İkisi de grid içerisinde değil ise grid döngüsü Şekil 5.7'deki gibi olacaktır.

Şekil 5.14'de örnek olarak sadece bir CR, FR ve SR eşleştirilmesi gösterilmiştir. Grid Kaynak sistemi bunun gibi uygun olan bütün kaynakları bulur ve maliyet olarak kullanıcının bütçesiyle uyan uygun kaynakların listesini gösterir. Aşağıda bununla ilgili olan kod parçası verilmiştir.

Pseudocode

```
CR List ← FindEligibleCR(Job)
İf(FR is in Grid System)
    FR List ← FindEligibleFR(Job)
İf(SR is in Grid System)
```

```
        SR List ← FindEligibleSR(Job)
While(CR in CR List)
 Begin
 CR Reserv List ← CheckRes(CR)
     CR Available List←FindEmptySlot(CR Reserv List,duration)
İf(CR Available List is not null)
       While(SR in SR List)
                     Begin
                   FR Reserv List ← CheckRes(FR)
                    FR Available List ← FindEmptySlot(FR Reserv
                                  List,CR Reserv List,duration)
                            İf(FR Available List is not null)
                           genList←AddGeneralList(CR Available
                                       List, FR Available List)
          End
        Else
            genList←AddGeneralList(CR Available List)
        End
CalculateCost(genList)
Display(genList)
```

5.2.3.6 Giriş dosyası belirli kaynaktan alınması ve çıkış dosyasının grid sistemde tutulması

Giriş dosyasının kullanıcı tarafından belirtilen kaynaktan alınması ve çıkış dosyasının ise grid sistem tarafından tutulması, işlemin yapılması için CR(İşlem Kaynağı) ve girdi dosyasının alınması için FR'a (Dosya Kaynağı) ve çıktı dosyasının saklanması için de SR'a (Veri Depolama Kaynağına) ihtiyaç duyulduğu anlamına gelmektedir. Giriş dosyasının sistemde bulunup bulunmamasına göre işlem yapılır. Eğer giriş dosyası için belirtilen kaynak sistemde var ise uygun alan Şekil 5.3'deki gibi olacaktır. Eğer sistem de yok ise uygun alan Şekil 5.9'daki gibi olacaktır. Algoritma aynı şekilde kalacaktır.

5.2.3.7 Giriş dosyasının grid sistemden alınması ve çıkış dosyasının yerel kaynakta tutulması

Giriş dosyasının grid sistem üzerinden alınması ve çıkış dosyasının ise yerel kaynakta tutulması, işlemin yapılması için CR(İşlem Kaynağı) ve girdi dosyasının

alınması için FR'a (Dosya Kaynağı) ihtiyaç duyulduğu anlamına gelmektedir. Grid döngüsü Şekil 5.11'deki gibi olacaktır. Burada yapılacak işlemin i/o olarak nekadar okuma/yazma yapacağı önemlidir. Eğer girdi dosyasının boyutu çıktı dosyasına oranla daha büyük ise bu durumda seçilecek CR ağ üzerinde geçen süreyi azaltmak adına FR'a yakın olacak şekilde seçilecektir. Bu durumdaki grid döngüsü Şekil 5.10'daki gibi olacaktır. Oradaki durumdan farklı olarak giriş dosyasını kendi kaynakları içerisinde arar ve FR listesini farklı kaynaklardan oluşturabilir. Algoritma aynı şekilde kalacaktır.

5.2.3.8 Giriş dosyasının grid sistemden alınması ve çıkış dosyasının grid sistem üzerinde tutulması

Giriş dosyasının Grid sistem üzerinden alınması ve çıkış dosyasının ise grid sistem üzerinde uygun olan herhangi bir kaynakta tutulması, işlemin yapılması için CR(İşlem Kaynağı) ve girdi dosyasının alınması için FR'a (Dosya Kaynağı) ihtiyaç duyulduğu anlamına gelmektedir. Burada yapılacak işlemin i/o olarak nekadar okuma/yazma yapacağı önemlidir. Eğer girdi dosyasının boyutu çıktı dosyasına oranla daha büyük ise bu durumda seçilecek CR ağ üzerinde geçen süreyi azaltmak adına FR'a yakın olacak şekilde seçilecektir. Grid Döngüsü Şekil 5.12'deki gibi uygun alan ise Şekil 5.13'deki ile aynı olacaktır.

Bu durumdaki algoritma diğer durumdan farklı olarak giriş dosyasını kendi kaynakları içerisinde arar ve FR listesini farklı kaynaklardan oluşturabilir.Algoritma aynı şekilde kalacaktır.

Pseudocode

```
CR List ← FindEligibleCR(Job)
İf(FR is in Grid System)
    FR List ← FindEligibleFR(Job)
İf(SR is in Grid System)
        SR List ← FindEligibleSR(Job)
While(CR in CR List)
Begin
CR Reserv List ← CheckRes(CR)
     CR Available List←FindEmptySlot(CR Reserv List,duration)
          İf(CR Available List is not null)
While(FR in FR List)
```

```
      Begin
FR Reserv List ← CheckRes(FR)
                    FR Available List ← FindEmptySlot(FR
              Reserv List,CR Reserv List,duration)
            İf(FR Available List is not null)
             While(SR in SR List)
      Begin
SR Reserv List ← CheckRes(SR)
                          SR Available List ← FindEmptySlot(FR
                                  ReservList,CR Reserv List,SR
                           Reserv List,duration)
                           genList←AddGeneralList(CR Available
                                      List, FR Available List,
                                            SR Available List)
End
           End
Else
             genList←AddGeneralList(CR Available List)
CalculateCost(genList)
Display(genList)
```

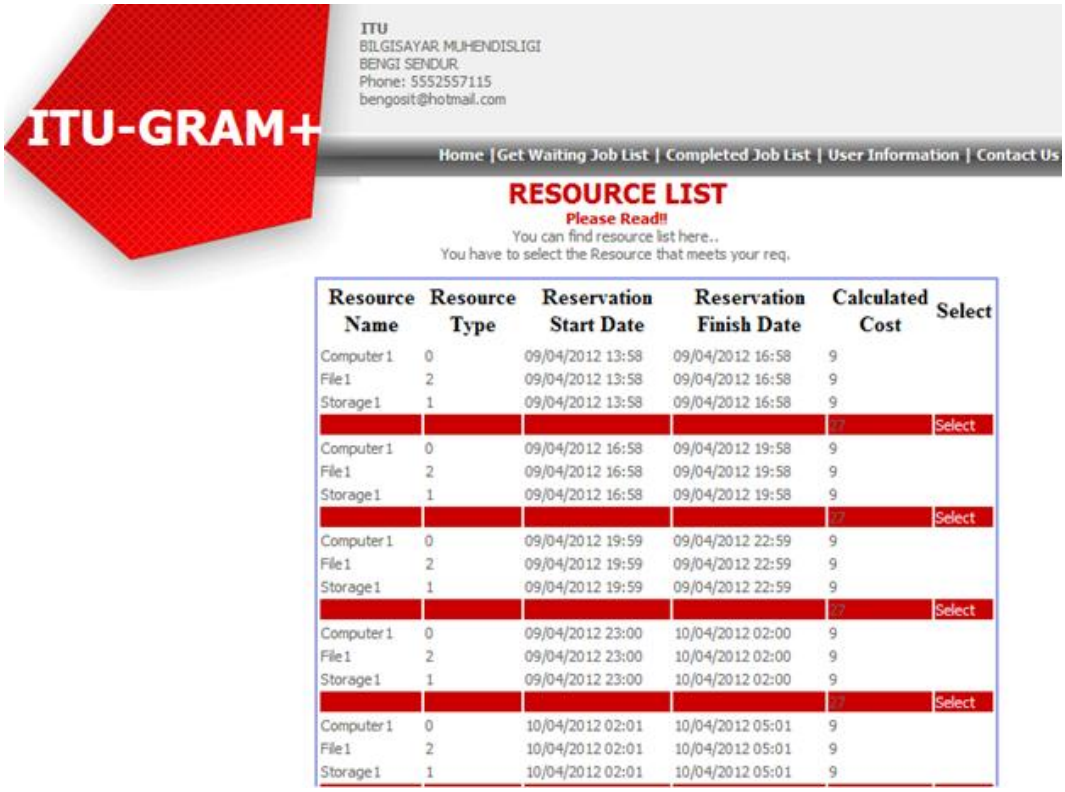
5.2.3.9 Giriş dosyasının grid sistemden alınması ve çıkış dosyasının belirli kaynak üzerinde tutulması

Giriş dosyasının Grid sistem üzerinden alınması ve çıkış dosyasının ise grid sistem üzerinde uygun olan herhangi bir kaynakta tutulması, işlemin yapılması için CR(İşlem Kaynağı) ve girdi dosyasının alınması için FR'a(Dosya Kaynağı) ihtiyaç duyulduğu anlamına gelmektedir. Burada yapılacak işlemin i/o olarak nekadar okuma/yazma yapacağı önemlidir. Eğer girdi dosyasının boyutu çıktı dosyasına oranla daha büyük ise bu durumda seçilecek CR ağ üzerinde geçen süreyi azaltmak adına FR'a yakın olacak şekilde seçilecektir.Grid Döngüsü Şekil 5.12'deki gibi uygun alan ise Şekil 5.13'deki ile aynı olacaktır. Bu durumdaki algoritma diğer durumlardan farklı olarak giriş dosyasını kendi kaynakları içerisinde arar.

FR listesini farklı kaynaklardan oluşturabilir.Algoritma aynı şekilde kalacaktır.

```
CR List ← FindEligibleCR(Job)
İf(FR is in Grid System)
    FR List ← FindEligibleFR(Job)
İf(SR is in Grid System)
        SR List ← FindEligibleSR(Job)
While(CR in CR List)
Begin
     CR Reserv List ← CheckRes(CR)
     CR Available List←FindEmptySlot(CR Reserv List,duration)
                    İf(CR Available List is not null)
       While(FR in FR List)
     Begin
FR Reserv List ← CheckRes(FR)
                  FR Available List ← FindEmptySlot(FR Reserv
                                      List,CR Reserv List,duration)
           İf(FR Available List is not null)
           While(SR in SR List)
      Begin
SR Reserv List ← CheckRes(SR)
                          SR Available List ← FindEmptySlot(FR
                                  ReservList,CR Reserv List,SR
                                         Reserv List,duration)
                           genList←AddGeneralList(CR Available
                                      List, FR Available List,
                                            SR Available List)
End
           End
        Else
            genList←AddGeneralList(CR Available List)
CalculateCost(genList)
Display(genList)
```

Şekil 5.15'de ITU-GRAM+ üzerinden alınan, kullanıcılara kaynak listesinin kombinasyonlarının sunulduğu bir ekran görüntüsüdür. Uygun kaynak listesi çok fazla olabilir bu durumda kullanıcıya gereksiz birçok alternatif sunmak yerine bütçesine uygun olan en fazla 10 kayıt gösterilir.



Şekil 5.15 : Kaynak listesi ekran görüntüsü.

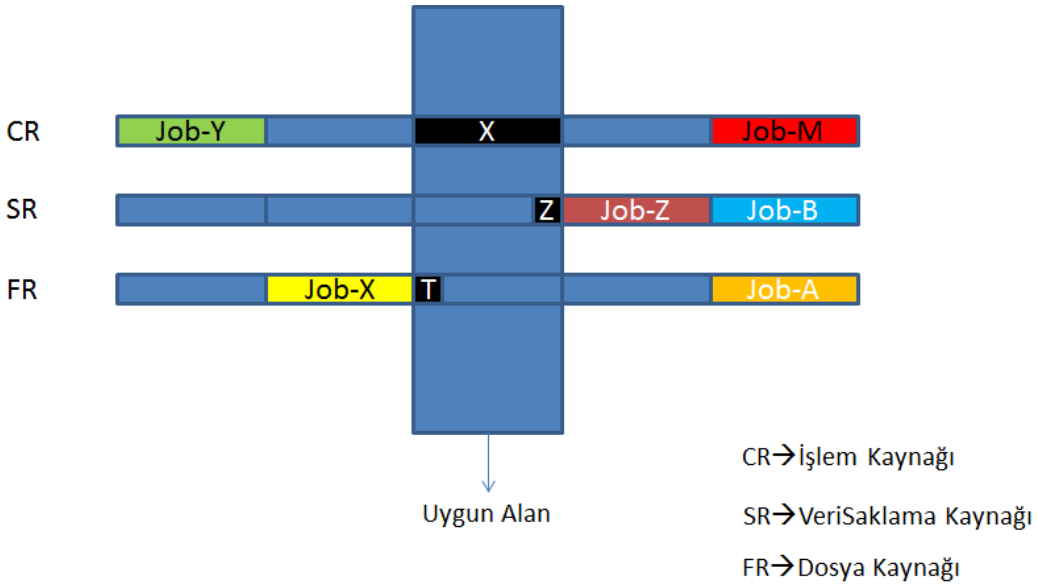
Genel İşleyiş

Sunucu üzerinde, daha önceden oluşturulmamış ise kullanıcı adı ile bir klasör oluşturulur. Bu klasör kullanıcının sistem üzerinde yapmış olduğu tüm işlerin bilgilerini tutar. Bir önceki adımda oluşturulan klasör altına verilen iş isminde yeni bir klasör oluşturulur ve kullanıcıdan alınan dosyalar bu klasör altına atılır. Kullanıcıdan alınan xml dosyası ayrıştırılır ve ekran üzerinden girilen parametreler ile birlikte Job sınıfından bir nesne oluşturulur. İş nesnesi içerisinde bulunan, kullanıcının girmiş olduğu minimum kaynak ihtiyaçlarına göre bu değerlere eşit veya bu değerlerden yüksek olan kaynaklar bulunur ve bir liste içerisine atılır. Kullanıcıdan alınan son teslim tarihi(deadline) bilgisine göre, işin gönderildiği tarih ile o tarih arasındaki zaman dilimi, işin çalışma süresi kadar zaman aralıklarına bölünür. Her bir zaman aralığı için bir önceki adımda oluşturulan liste içerisindeki kaynakların her biri için başka bir rezervasyonu olup olmadığı kontrol edilir. Kaynak üzerinde herhangi bir rezervasyon yok ise daha önceki adımlara bakarak finansal bir değer çıkarılır. Her kaynak sistem üzerinde tutulan ve bilgi güncellenmesi sonrası değişebilen bir ana maliyet değerine sahiptir. Bu değer üzerine maliyet hesaplama

bölümünde anlatılan işlemler yapıldıktan sonra bulunan değer eklenerek yeni bir değer ortaya çıkar. Son teslim tarihi ile rezervasyon tarihi arasındaki süre belirlenen sınır değerden yüksek ise kullanıcıya belirli oranda indirim yapılarak kaynak değeri hesaplanır. Kullanıcının daha önceki yaptığı işlem sayısına bakılarak belirlenen sınır değerinden düşük ise en son hesaplanan değer üzerinden indirim yapılarak son değer bulunur. Her bir rezervasyon aralığı için sistemdeki tüm kaynaklar üzerinde yüke bakılır. Eğer sistem yükü belirtilen değerin üstünde ise yeni bir yük getirmek maliyetli olacağından en son hesaplanan değer üzerine belirli oranda artırım yapılarak son değer hesaplanır. *Kaynak-Çalışma aralığı-Finansal değer* şeklinde bir liste oluşturulur ve veritabanına *suspended* durumunda kayıt edilerek, kullanıcının seçimine sunulur. Bir önceki adımda oluşturan liste kullanıcıların seçimine sunulur. Kullanıcılar kendi bütçesine uygun zaman aralığını seçer ve o an itibariyle o kaynak o kullanıcının işi için rezerve edilir.

5.2.4 Kaynak ayırma

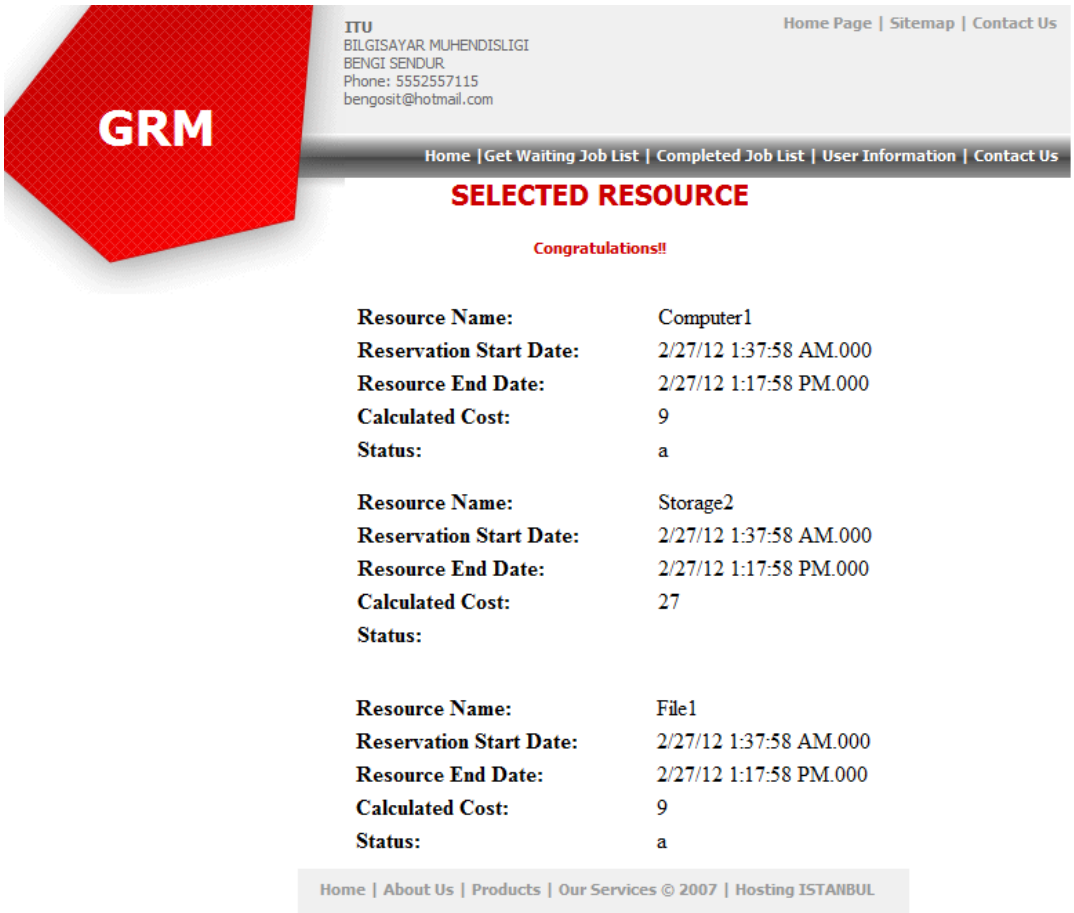
Kaynak ayırma işlemi,uygun kaynakların kullanıcıya listelenmesinden sonra kullanıcının bütçesine ve işlemine en uygun olan kaynak grubunu seçmesiyle başlar. Kullanıcıdan işlemin çalışma süresi ile ilgili bilgi alınır. Fakat bu süre içerisinde kullanıcının önceden tahmin edemeyeceği ağ üzerinde geçen bir süre vardır ve kaynak rezervasyonu yapılırken bu bilgiler de dikkate alınmalıdır. Aşağıda tüm kaynakların olduğu durum için bir örnek verilmiştir. Üç kaynak için de uygun olan bölge taranmıştır. Zaman çizelgesi üzerindeki T ve Z bölgeleri aslında o kaynağın gerçek anlamda ne kadar süre kullanılacağını gösterir. T ve Z bölgelerinin X bölgesine oranı grid tipine göre değişebilmektedir. Eğer data yoğunluklu bir grid ise bu durumda işin büyük bir kısmı SR ve FR üzerinde geçecek ,işlem yoğunluklu bir grid ise işin büyük bir kısmı CR üzerinde geçecektir. Giriş ve çıkış dosyaları işlem başlamadan önce kullanıcı tarafından verildiği için önce T süresi boyunca FR kullanılır ve dosya CR'a aktarılır.Bu süreden sonra FR'a ihtiyaç duyulmayacaktır. Dosya işlem süresi boyunca tekrar tekrar kullanılmak istendiğinde CR içerisinden yerel olarak dosyayı okuyacaktır. Aynı şekilde Z süresi boyunca oluşan çıkış dosyası kullanıcı tarafından belirtilen kaynağa transfer edilecektir.



Şekil 5.16 : Uygun alan içerisinde yeni iş tanımlama.

T= Network Bandwith/Expected Input File Size

Z= Network Bandwith/Expected Output File Size



Şekil 5.17 : Rezerve edilen kaynakların ekran görüntüsü.

Şekil 5.17'de ITU-GRAM+ üzerinden alınan bir ekran görüntüsü bulunmaktadır. Kullanıcı kendisine sunulan kaynak listesi içerisinde kendine en uygun olanını belirler ve uygulama üzerinden seçer. Kaynak listesi seçildikten sonra kullanıcıya hangi kaynak listesini hangi zaman aralıkları içerisinde seçtiğine dair ekrana bilgileri gönderir.

5.2.5 Kaynaklara gerekli dosyaları gönderme

Bu bölümde kullanıcıdan alınan uygulama dosyasının kaynak sisteme ne zaman ve nasıl gönderileceği anlatılacaktır.

İlk bölümde bahsettiğimiz en önemli problemlerden biri olan, ağ üzerinde dosya iletiminin, işi kaynağa gönderdikten sonra başlaması ve bu durumun o süre boyunca kaynağın gereksiz yere bekletilmesine yol açmaktadır.Bu durum,işi kaynağa göndermeden önce dosyaların kaynağa gönderilmesiyle çözülür.Bu işi yapan uygulama parçası aşağıdaki şekilde çalışır.

Pseudocode

```
Kaynak-Rez Liste←Durumu="Reserved" olan Kaynak-Rez Al.
     While(KR in Kaynak-Rez Liste)
Begin
      If((AktarılacakDosyaBoyutu/Bandwith)+Rezerv.Başlgç
                          Tarih>=Bugun)
 DosyaListesi=DosyalarıAl(KR.KullanıcıId,KR.İş Adı)
         While(Dosya in Dosya Listesi)
            Begin
              SendFTP(KaynakBilgileri,Dosya)
            End
    End
End
```

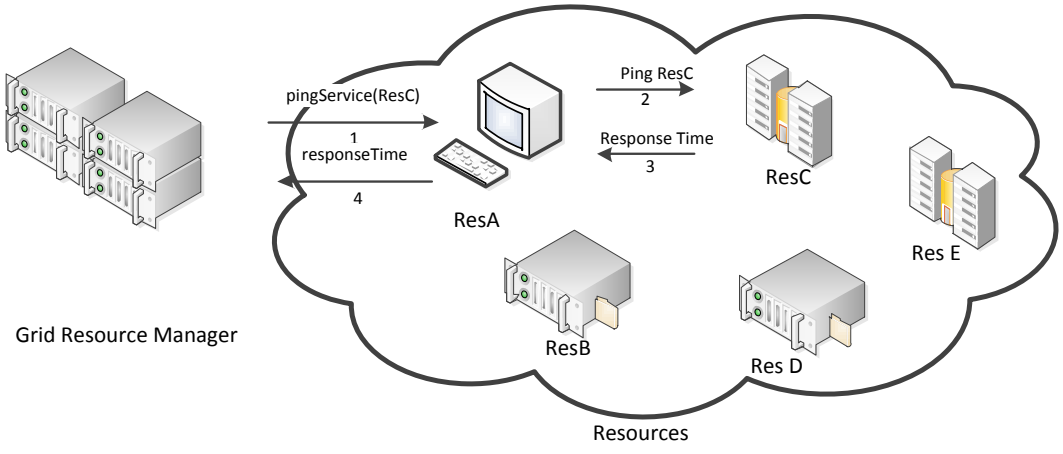
Quartz teknolojisi kullanarak bir zamanlayıcı ve bir de tetikleyici tanımlanmıştır. Zamanlayıcı sayesinde belirlenen zaman aralıkları ile bu uygulamayı çağıran sınıfın tetikleyici ile tetiklenmesi ile başlar. Durumu *active* olan işleri listeye alır. Bu işlere ait dosyaların büyüklüğünü hesaplar. Bu bilgi doğrultusunda dosyaların ağ üzerinde harcayacağı iletim sürelerini bulur. Bugünün tarihi üzerine, süre bilgisi ekler. Eğer elde edilen tarih işin başlama süresinden eşit veya fazla ise dosyaların iletmeye

başlar. İşin hangi kullanıcı tarafından gönderildiğine ve işin ismine bakarak dosya bilgilerine ulaşır. FTP protokolü ile kaynak üzerine ilgili dosyaları gönderir. Bir bilgilerine ulaşır.Tüm dosyaları alarak rezerve edilen işlem kaynağı ftp yaparak gönderir. Bu işlemler tamamlandıktan sonra kaynakların durumu *ready* olarak güncelenir.

5.2.6 Kaynakların maliyet hesaplaması

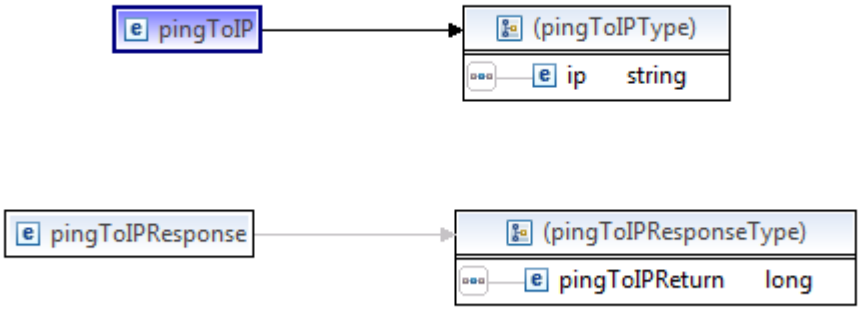
Kaynakların maliyeti, sahip oldukları donanım ve yazılımlar haricinde işin içeriği, çalıştırma zamanı, ağ maliyetleri gibi birçok değişken ile hesaplanmaktadır. Her bir kaynak dakika bazında bir birim maliyete sahiptir. Bu maliyetin üzerine işin çalıştıracağı andaki sistemdeki yüke, işin ne kadar zaman önceden sisteme giriş yapıldığına ve seçilen kaynakların birbirlerine olan uzaklıklarına göre ağ üzerindeki oluşabilecek maliyetler de eklenerek kullanıcıya en uygun olan kaynaklar listelenir. Kaynakların birbirlerine olan uzaklığını hesaplayabilmek için her kaynak üzerinde bir *Ping* servisi tanımlanmıştır.

Şekil 5.18'de işe uygun olan kaynakların birbirleri arasındaki uzaklıkların nasıl hesaplanacağını anlatmak için verilmiştir. Kaynak yönetim sistemi elinde farklı kombinasyonlardaki kaynaklar için birbirleri arasındaki uzaklıkları hesaplar. Bu hesaplamayı sadece ilk seferinde yapar ve veritabanına bu bilgileri kayıt eder. Bir sonraki eşleşmelerde ilk önce tabloyu kontrol eder eğer tabloda bu kaynaklar için herhangi bir veri girişi yapılmamışsa yukarıdaki şekilde gösterildiği gibi adımlardan geçer. Her bir kaynak kombinasyonu için yapılan bu hesaplamalardan sonra işin ağırlıklı olarak okuma veya yazma işlemi yapılacağı belirtilirse bu değerler de göz önüne alınarak bu kombinasyonlardan bazıları elenir.



Şekil 5.18 : Kaynaklar arasında çalışan ping servisi.

Şekil 5.18'de verilen işin daha çok yazma ağırlıklı bir iş olacağı belirtilmiş ise ;
Uzaklık(ResA-ResB)< Uzaklık(ResA-ResD)
olacağından uygun kaynak listesinden ResA-ResD eşleşmesi çıkarılır.
Eğer tam tersi olarak işin daha çok okuma ağırlıklı bir iş olacağı belirtilmiş ise;
Uzaklık(ResA-ResC)< Uzaklık(ResA-ResE)
olacağından uygun kaynak listesinden ResA-ResE eşleşmesi çıkarılır.
Bu uzaklığı hesaplayabilmek adına her bir kaynak üzerinde bir web servis tanımlanır. Bu web servis tanımı aşağıdaki şekilde gösterilmiştir.



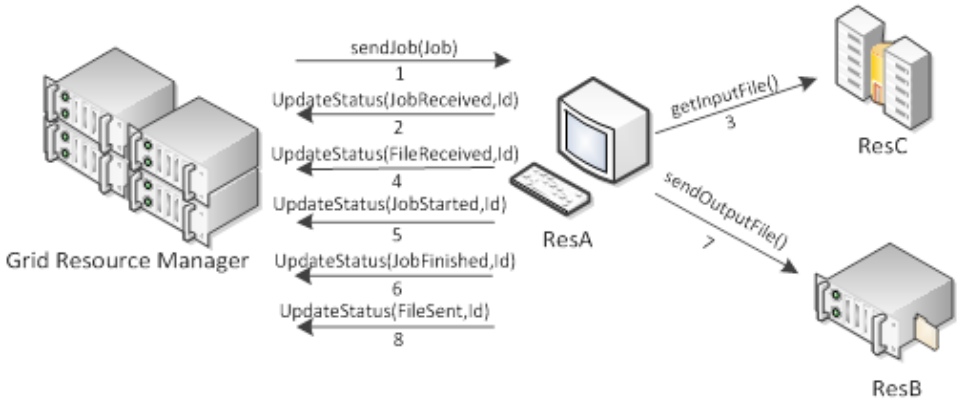
Şekil 5.19 : Ping servisi giriş ve çıkış bilgileri.

Kaynağa, bir diğer kaynağın ip adresi gönderilir ve dönüş olarak da milisaniye cinsinden bir değer alınır. Böylece kaynakların göreceli olarak birbirine göre uzaklıkları hesaplanmış olur.

5.2.7 Kaynaklara işi gönderme ve çalıştırma

Bu bölümde zamanı gelen işlerin kaynak sisteme nasıl gönderileceği ve iş sürecinin nasıl izleneceği anlatılacaktır.

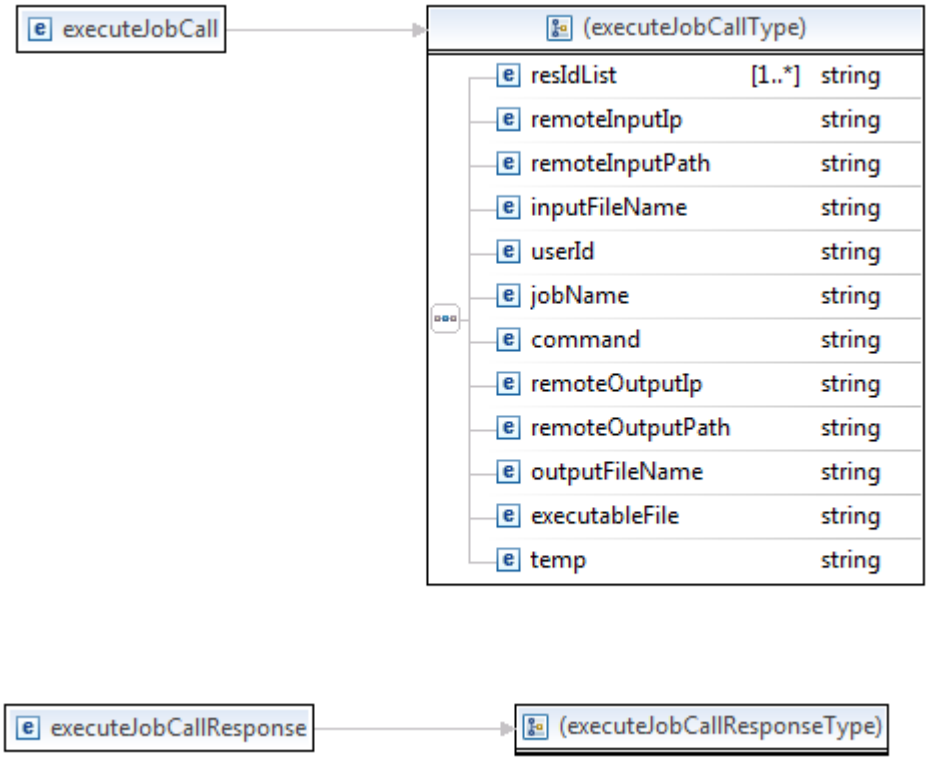
Tasarım bölümde bu işlemi karşılıklı olarak sunucu ve kaynak üzerinde web servisleri çağırarak çözeceğimizi belirtmiştik. Kaynak yönetim sisteminin doğrudan haberleştiği kaynak sadece CR olarak belirtilen işlem kaynağıdır. CR, isteği aldığına, girdi dosyasının sisteme ulaştığına, işlemin başladığına, işlemin bittiğine, çıktı dosyasının oluştuğuna ve çıktı dosyasının belirtilen kaynağa gönderildiğine dair kaynak yönetim sistemine bilgilendirmeler gönderir. İşlemin herhangi bir yerinde hata alırsa işlemi keser ve hata mesajı gönderir.Ya da kullanıcı belirli adımlarda işlemi iptal etmek isteyebilir.



Şekil 5.20 : Kaynağa işin gönderilmesi ve durum bilgisi alınması.

Bu uygulama parçacığı sırasıyla aşağıdaki adımları gerçekler.

1. Kaynak yönetim sistemi bu adımda işin çalıştırılması ve gerekli girdi ve çıktı dosyalarının nerede olduğunu ilgili tüm bilgileri ResA'a gönderir. Çağrılan web servisin girdi ve çıktı argümanlarının şemasal olarak gösterimi Şekil 5.20'de verilmiştir.

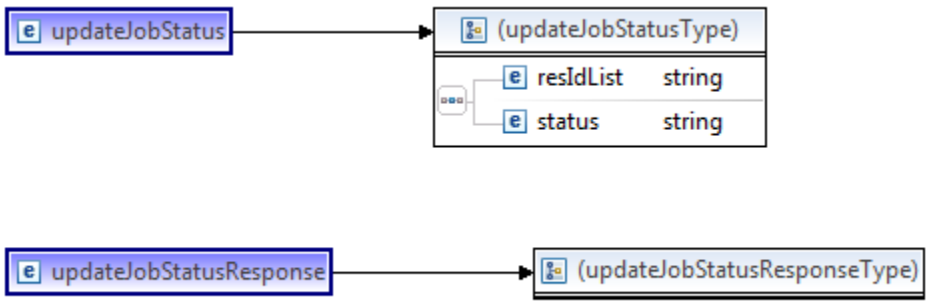


Şekil 5.21 : İş çalıştırma web servisi.

2. ResA isteğin ona ulaştığına dair güncelleme mesajı gönderir.

3. ResA işi çalıştırmadan önce gerekli girdi dosyasını ResC'den alır.
4. ResA girdi dosyasının geldiğine dair güncelleme mesajı gönderir.
5. ResA işi çalıştırmaya başlar ve çalışmaya başladığına dair güncelleme mesajı gönderir.
6. ResA üzerindeki iş sona erer ve bittiğine dair güncelleme mesajı gönderir.
7. Oluşan çıktı dosyasını ResB'ye gönderir
8. Dosyasının gönderildiğine dair bilgilendirme mesajı gönderir.

Tüm durum güncellemeleri için çağrılan web servisin girdi ve çıktı değerlerinin şemasal gösterimi Şekil 5.22'de verilmiştir.



Şekil 5.22 : Durum güncelleme web servisi.

Pseudocode(Sunucu)

```
Resource-Res List←Status="Ready"
While(KR in Kaynak-Res List)
   Begin
     New Thread(KR)
     ok=sendJob(KR,Job)
     if(ok)
         Status="Submitted"
    End
```

Pseudocode(Kaynak)

```
getFile(KR.Job)
Status="File Received"
If(command is  null)
     command=createCommand()
ok=ExecuteJob(command)
If(ok)
```

```
      Status=”Job Finished”
Else
      Status =”Failed”
sendFile(outputFile)
Status =”File Sent”
```

Genel İşleyiş

Quartz teknolojisi kullanarak bir zamanlayıcı ve bir de tetikleyici tanımlanmıştır. Zamanlayıcı sayesinde belirlenen zaman aralıkları ile bu uygulamayı çağıran sınıfın tetikleyiciyle tetiklenmesi ile başlar.

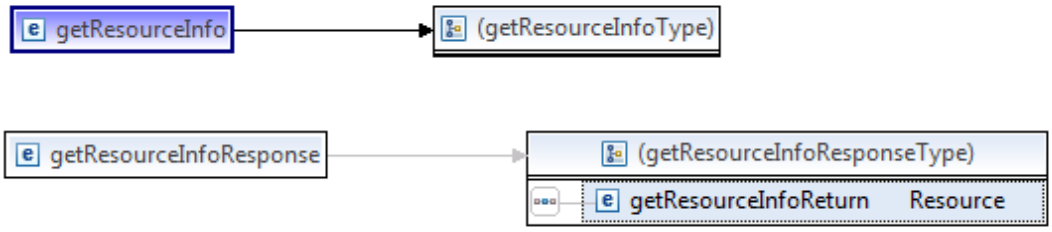
Durumu *ready* olan işlerden, işe başlama zamanı gelmiş olanları alır ve bir listeye yerleştirir. Listedeki her bir iş için yeni bir Thread başlatılır ve tüm işlerin paralel olarak çalışması sağlanır. Her bir thread, işe rezerve edilmiş olan kaynağın ip bilgilerini okur ve daha önceden kaynak tarafında gerçeklenmiş olan iş gönderme web servisi çağırılır. Bu servise işi çalıştırması için gereken bilgiler gönderilir.

Kaynak sistem bu web servis çağrısı kendisine ulaştığında, bilgilendirme olarak sunucu sisteme geri döner ve işin durumunu *received* olarak günceller. Kaynak sistem işe başlamadan önce gerekli dosyaların kendisinde bulunup bulunmadığını kontrol eder. Eğer dosyalar kendisinde varsa işe başlar ve sunucuya işe başladığına dair geri dönerek işin durumunu *started* olarak günceller.

Kaynak sistem işi bitirdiğinde sunucu sistemi işi bitirdiğine dair bilgi gönderir ve işin durumunu *completed* olarak günceller. Sunucu sistem bu bilgilendirmeyi aldıktan sonra kaynak sisteme FTP ile bağlanarak dosyaları kendine çeker ve kaynak sistem üzerinden dosyaları siler.

5.2.8 Kaynak bilgisini güncelleme

Bu bölümde kaynak sistem üzerindeki cpu ,storage ,network ,memory bilgilerinin sorgulanması ve belirli zaman aralıkları ile bu bilgilerin sunucu sistem üzerinde güncellenmesi işlemini içerir.
Şekil 5.23’de web servisin girdi ve çıktı argümanları şemasal olarak gösterilmiştir.
Girdi olarak herhangi bir değer gönderilmez sadece ilgili kaynağın ip bilgisi url içerisine eklenir ve çıktı olarak da kaynak üzerindeki fiziksel olarak gerekli bilgileri gönderir.



Şekil 5.23 : Kaynak bilgisi güncelleme web servisi.

Pseudocode

```
Kaynak Listesi←Tüm Kaynakları Al
While(Kaynak in Kaynak Listesi)
Begin
    EndPoint=Kaynak.ip
    if(WebService Çağır())
       Kaynak Bilgilerini Güncelle()
End
```

5.3 Durum Diyagramları

İşin rezervasyonun başlamasından işin kaynaklara gönderilip sonuçlarının gelmesine kadar geçen süre içerisinde birçok farklı duruma geçmektedir. Bu durumlardan bazıları sunucu sistem tarafından güncellenir bazıları ise iş başladıktan sonra kaynak üzerinden gelir. Bu durumların listesi ve birbirileri arasındaki geçişleri Şekil 5.24'de gösterilmiştir.

a→*Active* durumundan *Ready* durumuna uygulama dosyası CR'a gönderildiği zaman geçer.

b→*Ready* durumundan *Submitted* durumuna *sendJob*web servisi çağrıldığı zaman geçer.

c→*Submitted* durumundan *Started* durumunakaynak iş ile ilgili bilgileri aldığı zaman geçer.

d→*Started* durumundan *FileReceived* durumuna giriş dosyası geldiğini zaman geçer.

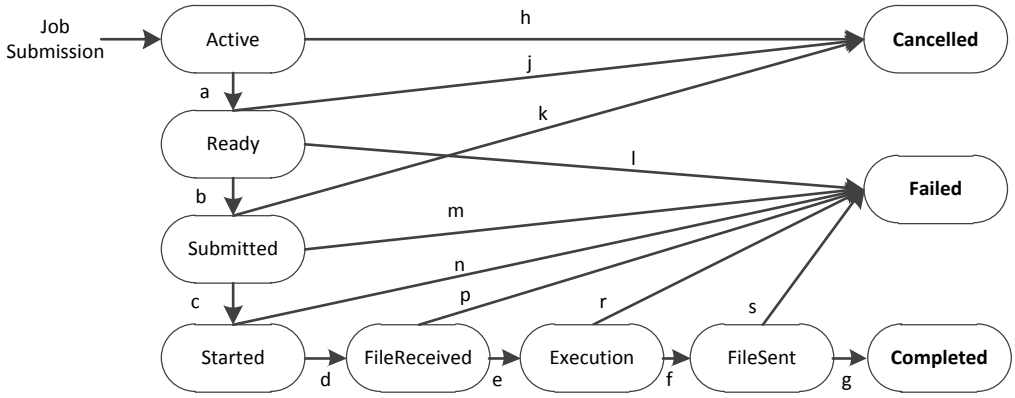
e→*FileReceived*durumundan*Execution*durumuna kaynak işi çalıştırmaya başladığı zaman geçer.

f→*Execution* durumundan *FileSent* durumuna çıkış dosyası gönderildiği zaman geçer.

g→*FileSent* durumundan *Completed* durumuna iş başarılı bir şekilde tamamlandığı zaman geçer.

h,j,k→*Active, Ready,Submitted* durumlarından *Cancelled* durumuna kullanıcı işi iptal ettiği zaman geçer.

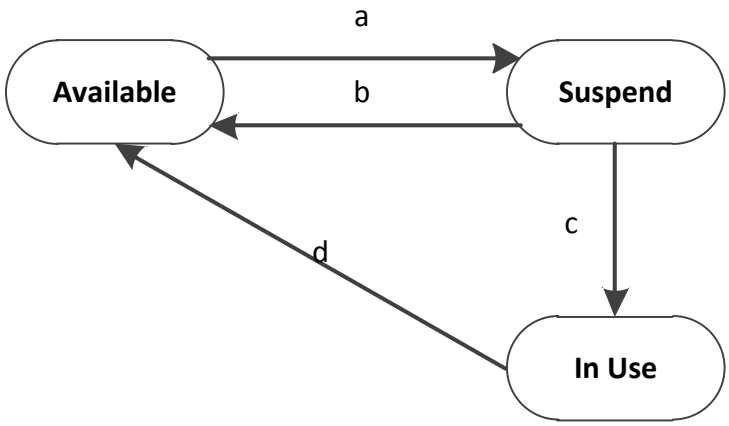
l,m,n,p,r,s→*Ready, Submitted, Started , FileReceived, Execution , FileSent* durumundan *Failed* durumuna bu süreçlerde herhangi bir hata oluştuğu zaman geçer.



Şekil 5.24 : İş durum diyagramı.

Şekil 5.24'de verilen durum diyagramının ilk adımı *Active* durumudur. Kullanıcı kendisine önerilen kaynak ve rezervasyon aralığından uygun olanını seçtikten sonra rezervasyon durumu *Active* olarak güncellenir. Bu durumda kaynak kullanıcıya rezerve edilmiştir ve işin bitiş anına kadar o kullanıcıya tanımlanmıştır. İş başlamadan önce gerekli dosyaların kaynak sisteme gönderilmesi için *Active* durumundaki rezervasyonlar alınır ve dosyalar kaynak sisteme iletilir. İletim işlemi tamamlandıktan sonra rezervasyonun durumu *Ready* olarak değiştirilir ve bunun da anlamı artık işin çalıştırabileceğidir. İşin başlama zamanı geldiğinde bu durumdaki tüm rezervasyonlar alınır ve kaynak sisteme işe başlayabileceğine dair bilgilendirme mesajı gönderilir ve rezervasyon durumu *Submitted* olarak güncellenir. Bu ana kadar tüm güncellemeler sunucu sistem tarafından yapılmaktadır. Bundan sonraki iş ile ilgili güncellemeler ise kaynak sistem tarafından yapılır. Kaynak sistem sunucundan bilgilendirmeyi aldığında iş ile ilgili gerekli dosyaların kendisinde olup olmadığını kontrol eder. İşe başlamak için herhangi bir engel bulunmuyorsa durumu *Started*

olarak günceller. Gerekli giriş dosyalarını diğer kaynak sistemden aldıktan sonra durumu *FileReceived* olarak güncellenir. Çalıştırma dosyasındaki ilgili kodları çalıştırmaya başlayınca durumu *Execution* olarak güncellenir. İş başarılı bir şekilde tamamlandıktan sonra dosyaları ilgili kaynak sisteme gönderir ve durumu *FileSent* olarak güncellenir. Tüm bu adımlar başarılı olarak gerçekleşir ise son adımda durum *Completed* olarak güncellenir. Kullanıcı işi kendisi iptal etmek ister ise durumu *Cancelled* olarak değiştirilir. Eğer adımların herhangi birisinde işlem başarısız olur ise durum *Failed* olarak güncellenir. Şekil 5.25'de ise kaynakların durumları ve arasındaki geçişler ile ilgili diyagram verilmiştir.



Şekil 5.25 : Kaynak durum diyagramı.

Her kaynak ilk öncelikle *Available* durumdadır. Herhangi bir iş için rezervasyon yapılmamıştır. Bu durumdan *Suspend* durumuna geçebilir. *Suspend* durumu web kullanıcıları arasında aynı kaynağın aynı anda rezerve edilmesini önlemek amacıyla oluşturulmuştur ve geçici bir durumdur. *Suspend* durumunda olan bir kaynak ya *In Use* durumuna yada tekrar *Available* durumuna geçer. *In Use* kaynağın bir iş için rezerve olduğunu gösterir. İş tamamlandıktan sonra kaynak tekrar *Available* durumuna geri döner.

6. SONUÇ

ITU-GRAM+, kullanıcılara işleri koşturabilmeleri için standart arayüz ve protokoller sağlayan bir ara katman yazılımıdır. En belirgin özelliklerinden birisi sadece açık kaynak kodlu yazılımlar kullanılarak gerçeklenmiş olmasıdır. Gerçeklenen sistemde, kaynakların bulunması ve kullanıcılara önerilmesi aşamasında birden çok iş parametresi göz önünde bulundurulmuş ve hem sistemin performansının en iyi şekilde kullanılabileceği hem de her kullanıcının bütçesi dahilinde en iyi hizmeti alabileceği bir sistem tasarlanmıştır. Bu sistemde kullanıcılara standartlar arayüzler sağlanmış ve kullanıcıyı gereksiz teknik ayrıntılar ile uğraşmaktan uzaklaştırmıştır. Herhangi bir yazılım ve bilgisayar bilgisine sahip olmayan kullanıcının bile kolay bir şekilde işlerini yürütebileceği bir ortam sağlanmıştır. Aynı zamanda kullanıcılara kaynakları istediği zaman için rezervasyon yapabilme imkanı sunulmuş ve böylece kaynakların hem sistem tarafından hem de kullanıcı tarafından daha verimli kullanılması sağlanmıştır. Kaynaklar arasındaki iletişimi web servisler aracılığıyla yaparak hem yeni kaynakların sisteme katılmalarını kolaylaştırmış hem de farklı özellikteki her bir kaynak için ayrı bir arayüz yapılmasına gerek kalmamıştır. Yazılım geliştiriciler bakış açısıyla bakıldığında onlara açık kaynak kodlu yazılımlar sunarak kendi ara katman yazılımlarını oluşturabileceği ve geliştirebileceği bir ortam verilmiştir. ITU-GRAM+ ‘ın kullandığı teknolojiler göz önüne alındığında kurulumu ve kullanıma başlanması hızlı ve kolay bir şekilde yapılabilmektedir.

ITU-GRAM+ geliştirilirken basit güvenlik mekanizmaları düşünülerek ilerlenmiş fakat bunun için ayrı bir bileşen tanımlanıp, geliştirilmemiştir. ITU-GRAM+ ‘ın sağlamış olduğu teknoloji ve altyapı ile gelişime ve değişime açıktır. Bunun yanı sıra işin çalışması sırasında oluşabilecek hatalar sonucu işlerin tekrar sıralanması kapsam dışında bırakılmıştır ve gelecek çalışmalara altyapı sağlamıştır.

KAYNAKLAR

[1]Buyya, R. ve Venugopal, S(2005). A Gentle Introduction to Grid Computing andTechnologies, CSI Communications, India, 29(1)

[2]Foster, I (2002)What is the Grid? A Three Point Checklist, Grid Today, vol.16.

[3]Foster, I. ve Kesselman, C. ve Tuecke, S.(2001). The Anatomy of Grid: EnablingScalable Virtual Organizations,International Journal of SupercomputerApplications

[4]Fundamentals of Grid Computing. Alındığı tarih: 30.04.2012, adres: http://www.redbooks.ibm.com/

[5]Krauter,K. ve Buyya, R. ve Maheswaran,M. (2002) A Taxonomy and Survey of GridResource Management Systems for Distributed Computing, Software:Practice and Experience (SPE), ISSN 0038-0644 32(2) Wiley Press, NewYork, USA, Pp. 135-164.

[6]Schopf, JM.(2004)Ten actions when Grid scheduling: the user as a Grid schedulerKluwer Academic Publishers Norwell, MA, USA ©2004 table of contents ISBN:1-4020-7575-8

[7]Job Submission Description Language (JSDL) Specification Alındığı tarih: 30.04.2012, adres:http://www.gridforum.org/documents/GFD.56.pdf

[8]Feller, M. ve Foster, I. ve Martin, S.(2007) GT4 GRAM: A Functionality andPerformance Study, TERAGRID Conference, Madison

[9]Foster, I. ve Kesselman, C. ve Nick, M. ve Tuecke,S.(2002) The Physiology of the Grid.

[10]Foster,I. (2005) Globus Toolkit Version 4:Software for Service-Oriented System

[11]Frey, J. Ve Tannenbaum, T. veFoster, I. ve Livny, M. ve Tuecke, S.(2001) Condor-G: A Computation Management Agent for Multi-Institutional Grids. Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10) San Francisco, California, August 7-9.

[12]Natrajan,A. ve Nguyen-Tuong, A. Ve Humphrey, M. ve Herrick, M. ve Clarke,B. P. ve Grimshaw, A. S. (2002).The Legion Grid Portal. Concurrency and Computation: Practice and Experience 14(13-15): 1365-1394

[13]The Legion Group(2001). Legion 1.8 Basic User Manual

[14]Mysql, Alındığı tarih: 30.04.2012, adres: http://www.mysql.com/

[15]Java,Alındığı tarih: 30.04.2012, adres:http://www.oracle.com/technetwork/java/

[16]Hibernate, Alındığı tarih: 30.04.2012, adres: http://www.hibernate.org/

[17]Struts2, Alındığı tarih: 30.04.2012, adres: http://struts.apache.org/2.x/

[18]Apache Tomcat, Alındığı tarih: 30.04.2012, adres:http://tomcat.apache.org/

[19]Axis2,Alındığı tarih: 30.04.2012,adres: http://axis.apache.org/axis2/java/core/

[20]Ant,Alındığı tarih: 30.04.2012,adres: http://ant.apache.org/

[21]Log4j,Alındığı tarih: 30.04.2012,adres:http://en.wikipedia.org/wiki/Log4j

[22]Quartz Scheduler,Alındığı tarih: 30.04.2012,adres: http://quartz-scheduler.org/

[23]CSS,Alındığı tarih: 30.04.2012,
adres: http://tr.wikipedia.org/wiki/Cascading_Style_Sheets

[24]JavaScript,Alındığı tarih: 30.04.2012,
adres: http://tr.wikipedia.org/wiki/JavaScript

[25]UML,Alındığı tarih: 30.04.2012,adres: http://tr.wikipedia.org/wiki/UML

EKLER

EK A: Kurulum dosyası

EK A

GELİŞTİRME ORTAMININ KURULMASI

Eclipse Kurulumu

Eclipse'in Ganymede versiyonu kullanılmıştır. Aşağıdaki linkten indirilebilir.
http://www.eclipse.org/ganymede/
Kurulumu yapıldığı klasör altında eclipse.exe çalıştırılarak uygulama ortamı açılır. Açılırken uygulama kodlarının bulunduğu workspace seçilir. Ekte verilen war uzantılı dosya proje olarak import edilerek uygulama kodlarına ulaşılır. 3 farklı proje bulunmaktadır. Bunlardan ikisi sunucu tarafı birisi kaynak tarafında koşturulacaktır.
Sunucu tarafındaki projerin isimleri aşağıdaki gibidir.
GridResourceManager: Web uygulamasının bulunduğu projedir. Sunucu üzerinde çalıştırılacaktır. Uygulamaya ait konfigürasyon dosyaları aşağıda verilmiştir.
Config.properties: Uygulama içerisinde kullanılan konfigüratif bilgiler bu dosya içerisinde yer almaktadır.Dosyadaki her bir değişkenin nerede kullanıldığı yorum satırı olarak verilmiştir.
Hibernate.cfg.xml: Veritabanı bağlantı bilgilerinin ve veritabanı nesnelerinin yer aldığı konfigürasyon dosyasıdır.
Log4j.properties: Uygulamanın loglama ile ilgili bilgilerinin yer aldığı konfigürasyon dosyasıdır.
Build.xml: Uygulamanın nereye deploy edileceği ile ilgili bilgileri içeren konfigürasyon Apache Ant dosyasıdır.Proje build edildikten sonra bu dosya execute edilerek Tomcat altından belirtilen isim ile .war dosyasının oluşmasını sağlar.
Struts.xml:Action sayfalarının tanımlandığı dosyadır.Hangi Action'da sayfanın hangi sayfaya yönlendirileceği ile ilgili bilgileri içerir.

GridJobStatusUpdateService: İş ile ilgili durum bilgilerini almak için yazılmış bir web servis projesidir. Sunucu tarafında çalıştırılacaktır. Aşağıdaki konfigürasyon dosyaları bulunmaktadır.
Hibernate.cfg.xml: Veritabanı bağlantı bilgilerinin ve veritabanı nesnelerinin yer aldığı konfigürasyon dosyasıdır.
Log4j.properties: Uygulamanın loglama ile ilgili bilgilerinin yer aldığı konfigürasyon dosyasıdır.

Build.xml: Uygulamanın nereye deploy edileceği ile ilgili bilgileri içeren konfigürasyon Apache Ant dosyasıdır.Proje build edildikten sonra bu dosya execute edilerek Tomcat altından belirtilen isim ile .war dosyasının oluşmasını sağlar.

Apache Tomcat Kurulumu

Apache Tomcat v5.5 kullanılmıştır.Aşağıdaki linkten indirilebilir.
http://tomcat.apache.org/download-55.cgi
Kurulumun yapıldığı klasör altından /bin/start.bat dosyası çalıştırılarak Tomcat'in başlaması sağlanır.Geliştirme ortamı üzerinde http://localhost:8080/ yazılarak Tomcat ana sayfasının geldiği görülünce Tomcat Sunucusunun çalıştığı görülür. Eclipse'den de ayrıca Server konfigürasyonu yapılarak Tomcat klasörü gösterildiği taktirde Eclipse üzerinden de start/stop komutları verilebilir. Uygulamanın sunucu server üzerinde çalışabilmesi build edilip paketlenen war dosyası /webapps altına koyulur.Sunucu çalıştığı anda bu dosya sunucu server tarafından extract edilerek klasör oluşturulur.
Tüm konfigürasyon dosyaları /conf altında bulunmaktadır. Konfigürasyon dosyalarında herhangi bir değişiklik yapılmasına gerek yoktur.

MYSQL Kurulumu

MYSQL'in MySQL Server 4.1 versiyonu kullanılmıştır.Aşağıdaki linkten indirilebilir.
http://dev.mysql.com/downloads/mysql/4.0.html
MYSQL için özel bir arayüz kullanılmamış fakat Eclipse içerisinde CLAY plugin'i eklenmiştir.Bu plugin sayesinde grafik arayüzü ile veritabanı tabloları oluşturulup create scriptleri alınabilmektedir.Bu scriptler MYSQL'in konsol uygulaması(mysql.exe) çalıştırılıp ilgili tablolar oluşturulabilmektedir.

Hibernate Kurulumu

Hibernate Framework'u genelde Eclipse içerisinde bundle olarak gelmektedir.Aşağıdaki linkten de indirilebilir.
http://www.hibernate.org/downloads
Hibernate için konfigürasyon dosyası yukarıda projeler tanıtılırken verilmiştir.

Struts2 Kurulumu

Struts2 Framework'u genelde Eclipse içerisinde bundle olarak gelmektedir.
Aşağıdaki linkten de indirilebilir.
http://struts.apache.org/download.cgi
Struts2 için konfigürasyon dosyası yukarıda projeler tanıtılırken verilmiştir.

Axis-2 Kurulumu

Axis2 eklentisi Eclipse içerisinde bundle olarak gelmektedir.Aşağıdaki linkten de indirilebilir.

http://axis.apache.org/axis2/java/core/download.cgi

KAYNAKLARA ORTAMIN KURULMASI

Kaynak tarafındaki projelerin isimleri aşağıdaki gibidir.
GridWebService: Kaynak tarafında çalıştırılacaktır. İçerisinde hem diğer kaynaklar hem de sunucu ile haberleşmesini birçok metot bulunmaktadır. Aşağıdaki konfigürasyon dosyaları bulunmaktadır.
Config.properties:Uygulama içerisinde kullanılan konfigüratif bilgiler bu dosya içerisinde yer almaktadır.Dosyadaki her bir değişkenin nerede kullanıldığı yorum satırı olarak verilmiştir.
Log4j.properties:Uygulamanın loglama ile ilgili bilgilerinin yer aldığı konfigürasyon dosyasıdır.
Build.xml:Uygulamanın nereye deploy edileceği ile ilgili bilgileri içeren konfigürasyon Apache Ant dosyasıdır.Proje build edildikten sonra bu dosya execute edilerek Tomcat altından belirtilen isim ile .war dosyasının oluşmasını sağlar.
Tomcat Server Kurulması

Apache Tomcat v5.5 kullanılmıştır.Aşağıdaki linkten indirilebilir.
http://tomcat.apache.org/download-55.cgiKurulumun yapıldığı klasör altından /bin/start.bat dosyası çalıştırılarak Tomcat'in başlaması sağlanır.Geliştirme ortamı üzerinde http://localhost:8080/ yazılarak Tomcat ana sayfasının geldiği görülünce Tomcat Sunucusunun çalıştığı görülür.Eclipse'den de ayrıca Server konfigürasyonu

yapılarak Tomcat klasörü gösterildiği taktirde Eclipse üzerinden de start/stop komutları verilebilir. Uygulamanın sunucu server üzerinde çalışabilmesi build edilip paketlenen war dosyası /webapps altına koyulur.Sunucu çalıştığı anda bu dosya sunucu server tarafından extract edilerek klasör oluşturulur.

FTP Server Kurulumu

FTP Server olarak Filezilla seçilmiştir.Dosya alıp göndermek için kaynaklara kurulması gerekmektedir.Aşağıdaki linkten download edilebilir.

http://filezilla-project.org/download.php?type=server

ÖZGEÇMİŞ

Ad Soyad:Gülden Bengi ŞENDUR

Doğum Yeri ve Tarihi: Kelkit, 15.04.1986

Adres: Yeşil Park Evleri C Blok D:21 Umraniye/Istanbul

E-Posta: bengosit@hotmail.com

Lisans: İstanbul Universitesi/Bilgisayar Mühendisliği

Mesleki Deneyim ve Ödüller: Avea-CRM Uzmanı

Yayın ve Patent Listesi:

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Sendur G.B. and Altılar D.T. , 2012:** ITU-GRAM+: A WEB Based Grid Middleware. WorldComp'12 The 2012 *International Conference on Grid Computing and Applications*, July 16-19, 2012 Las Vegas, USA.