Computação em Grid

Otavio Rodolfo Piske – angusyoung@gmail.com
Especialização em Software Livre

Centro Universitário Positivo – UnicenP

1.Índice

Resumo

Este artigo faz uma explicação inicial sobre as várias tecnologias relacionadas a computação distribuída para então abordar mais a fundo a respeito da tecnologia de computação em grid. Por fim, o artigo descreve 2 conhecidas ferramentas de código aberto utilizadas para desenvolvimento e implementação de sistemas grids.

Abstract

This article initiates explaining about the various distributed computing related technologies It then, details about grid computing technology. Finally it describes 2 well known open source toolkits used to develop and implement grids systems.

1. Introdução

Computação em Grid é o termo utilizado para se referir a uma técnica computacional que utiliza os recursos de diferentes computadores com o intuito de resolver problemas de grande complexidade e/ou volume, estando portanto, não limitada apenas à execução distribuída de algoritmos de processamento, mas também gerenciamento de grande quantidade de dados distribuídos. Estes não precisam necessariamente ser o resultado prático da execução de um grid computacional.

Indo ainda mais a fundo na definição de Grid, é possível utilizar-se da definição de Buya[1]: “A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements” (Trad..: “Um tipo de sistema distribuído e paralelo que possibilita o compartilhamento, seleção e agregação dinamicamente, em tempo de execução, de recursos autônomos geograficamente distribuídos, de acordo com a sua disponibilidade, capacidade, performance, custo e requerimentos, do usuário, de qualidade de serviço).

De certa maneira, a computação em Grid encontra-se na mesma área de atuação dos clusters e supercomputadores (muitas vezes atuando em conjunto com ambos).

Com o poder de processamento dos computadores atuais aumentando a cada ano, aumenta a complexidade (e o conjunto) de problemas que podem ser resolvidos utilizando-se de grids.

1.1. Computação Paralela

Computação paralela é uma técnica de programação cujo objetivo é executar operações em paralelo, desta maneira obtendo maior desempenho em sistemas multiprocessados, grids, etc.

De modo geral o trabalho de tornar um programa capaz de ser executado em paralelo consiste basicamente em quebrar suas tarefas em partes menores. É importante notar, porém, que nem todos os programas podem ser otimizados desta maneira.

1.2. Computação Distribuída

Computação distribuída é um sub-ramo da computação paralela na qual parte das operações ocorre em máquinas diferentes daquela executando o fluxo principal de um programa.

Duas das propriedades da computação distribuída são:

- abertura: referente à capacidade de cada subsistema estar continuamente aberto à interação com outros subsistemas.
- escalabilidade: referente à capacidade de cada subsistema ser expandido administrativa, geográfica e localmente.

A maneira como cada propriedade da computação distribuída é aplicada é variável de acordo com a arquitetura utilizada.

1.2.1. Computação Distribuída: Arquitetura

O conceito de Computação Distribuída é bastante amplo, deste modo, é conveniente dividi-lo em arquiteturas visando um melhor entendimento, bem como endereçando suas particularidades em conceitos distintos.

Imagem gentilmente cedida por www.gridcafé.org

Entre as arquiteturas de computação distribuída existentes é importante citar:

- Cliente/Servidor: arquitetura na qual uma parte do processamento é executada no cliente e a outra parte é

executada no servidor, podendo o servidor atender 1 ou mais clientes simultaneamente. Ex.: servidor ftp e cliente ftp.

- 3-Camadas: arquitetura semelhante à cliente/servidor, entretanto à lógica específica da aplicação é executada em um agente intermediário.
- Peer To Peer: arquitetura na qual não existe um agente responsável por gerenciar ou prover recursos. Neste caso estas responsabilidades são divididas entre todos os agentes da rede.

Entretanto, essas 3 arquiteturas não são as únicas existentes, embora sejam as mais importantes e conhecidas. Para a completude do artigo é conveniente citar, também, a existência das seguintes arquiteturas: n-camadas, orientada à serviço, código móvel, repositório replicável, etc.

1.3.Um Pouco de História

Embora o Grid como este é concebido atualmente seja uma idéia relativamente nova, os conceitos nos quais esta tecnologia se baseia não são tão novos assim e já fazem parte da história da computação à algum tempo.

Como exemplo disso pode-se citar o conceito de compartilhamento de processamento, muito popular nas décadas de sessenta e setenta, quando a capacidade de processamento dos mainframes ainda era muito limitada em relação à sua aplicação.

Outro conceito importante é baseado na idéia de meta computação (metacomputing), idéia esta, popular na década de 90, e consistia em compartilhar processamento através de centros de supercomputadores.

Por volta desta mesma época 2 projetos recém iniciados influenciaram muito o design e aplicação dos Grids atuais:

- FARNER (Factory via Network Enable Recursion): inspirou conceitos de quebra e distribuição de grandes problemas computacionais.
- I-Way (Information Wide Area Year): projeto cujo objetivo era ligar redes de supercomputadores. Inovou através da utilização de “resource brokers”, os quais eram conceitualmente parecidos com os “resource brokers” utilizados atualmente.

Adiciolnamente, é interessante notar que, em geral, um computador atual é tão potente quanto um gigantesco supercomputador de uma década atrás, tornando o conceito de meta computação ultrapassado.

1.4. Conceitos

Assim como nos clusters, muito do poder computacional de um Grid está dívido em diversas máquinas (também chamadas de membros). Desta forma é possível assegurar que o trabalho de um sistema Grid não está relacionado apenas ao processamento de dados, mas também ao gerenciamento dos recursos alocados ao sistema. Por fim, isto torna possível dividir o funcionamento básico de um Grid em camadas: camada de rede, camada de recursos, middleware, aplicação e serviços.

Imagem gentilmente cedida por www.gridcafé.org

Esta divisão além de tornar mais simples o entendimento do funcionamento de um grid, torna possível a criação de grids de propósito geral. Grids de propósito geral são um tipo específico de implementação de grid em que a parte utilizada para o processamento do dados está separada da parte utilizada para gerenciar o funcionamento do grid.

Uma vez que um grid é constituído de membros heterogêneos é fácil perceber que a interoperabilidade é um conceito chave no funcionamento de um Grid. Em geral os Grids resolvem este problema através da utilização de protocolos abertos (TCP, UDP, IP, Globus Toolkit, etc). Isto acaba levando alguns dos teóricos, como é o caso de Foster[2], considerarem este como um conceito chave no funcionamento e implementação de grids.

Ainda, o CERN[9], define 5 conceitos básicos para a definição de um cluster:

- Compartilhamento de recursos: refere-se ao compartilhamento de recursos computacionais.

Imagem gentilmente cedida por www.gridcafé.org

- Uso de recursos: refere-se ao uso eficiente dos recursos disponíveis e está ligado ao princípio da alocação eficiente de recursos.
- Acesso seguro: devido a necessidade de garantir a confiabilidade e a segurança dos dados, um cluster deve endereçar os problemas inerentes à política de acesso, autorização e autenticação. No caso de grids comerciais este conceito é ainda mais importante, pois é através dele que é definido quem usa o que.
- Morte da distância: refere-se a insignificância da distância entre os membros do grid.
- Padrões abertos: garante a comunicação plena dos membros do grid através da utilização de padrões abertos. Estes padrões vêm sendo definidos atualmente através de uma entidade chamada Global Grid Forum.

1.5. Arquitetura

Como mencionado anteriormente, a arquitetura de um grid pode ser dividida em arquiteturas, visando facilitar seu design e entendimento. Entretanto não existe apenas uma maneira de descrever um grid em camadas, assim sendo será analisado as duas mais comuns.

Uma maneira mais simples de descrever a arquitetura de um grid baseia-se na divisão do mesmo em camadas de acordo com o seu recurso.

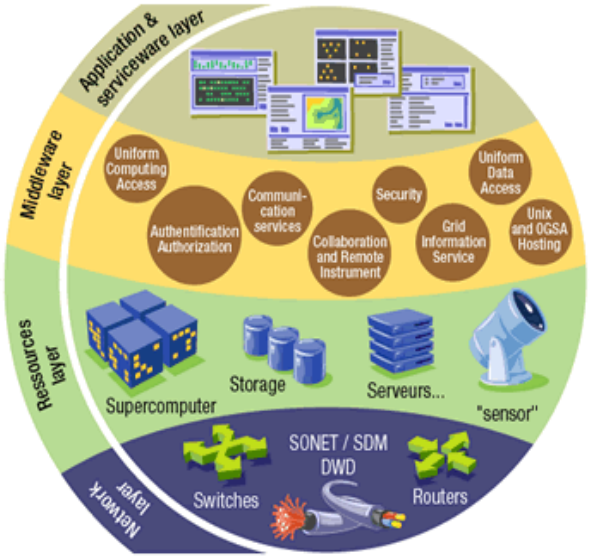
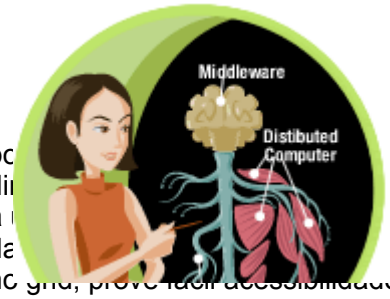


Imagem gentilmente cedida por www.gridcafé.org

Desta maneira tornando possível dividir um Grid seguintes quatro camadas:

- Rede: define a conectividade entre os membros do grid, e pode ser considerado o sistema nervoso de um grid. É interessante notar, também, que na grande maioria das vezes os sistemas em grids utilizam os mesmos tipos de links disponíveis para qualquer usuário comum: links internet, ethernet 10/100/1000Mbps, etc.
- Recursos: define os recursos membros do grid, como computadores, sistemas de armazenamento, sensores, etc.
- Middleware: responsável pela interconectividade entre os recursos do grid, bem como a segurança dos dados e comunicação, etc. Entre suas funções também pode-se citar as negociações máquina-a-maquina (M2M – Machine 2 Machine). Esta camada é, muitas vezes, constituída de um grande conjunto de softwares. Como exemplo disso, é possível citar o projeto europeu de grid de dados: European Data Grid, o qual é constituído de aproximadamente 300 mil linhas de código fonte. Muitos desses softwares atuam negociando transações de dados e outros recebendo e gerenciando-os. Fazendo, novamente, uma análogia com relação ao corpo humano, pode-se dizer que a camada Middleware é o cérebro do Grid.



po o
di , negocia
a nto de
da oblemas
no grid, provê fácil acessibilidade a dados críticos através da sua replicação, provê políticas de acesso ao grid.

- Aplicação: aplicações que rodam no grid. É a responsável por obter as credenciais de segurança necessárias para obtenção dos dados, negocia a obtenção de dados com a camada de serviços coletivos, monitorar o progresso das requisições, processamento e transferências de dados.

- Aplicação e Serviços: aplicações (científicas, econômicas, de engenharia, etc) que rodam no grid, ferramentas de desenvolvimento, portais, etc.

Entre os experts em grids, porém, não é incomum encontrar os grids definidos conforme a sua estrutura física (hardware, redes, aplicação etc). Embora um pouco mais complicada de ser entendida por pessoas com pouca ou nenhuma experiência com tecnologia em geral, esta fornece uma visão ainda mais clara sobre as n-camadas que compõem um Grid. Esta definição pode ser melhor entendida no gráfico abaixo:

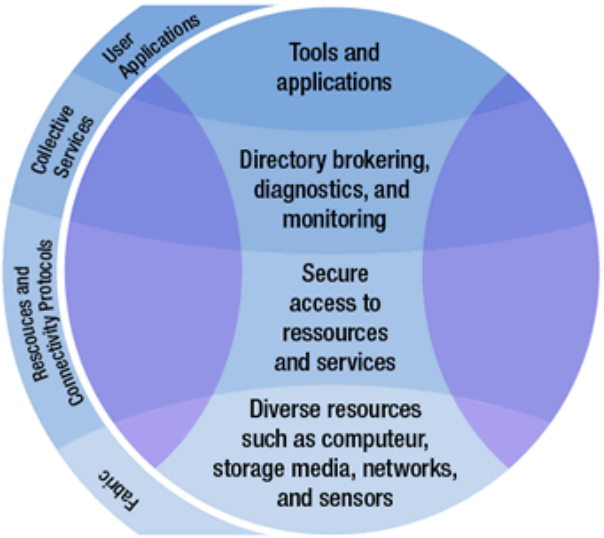


Imagem gentilmente cedida por www.gridcafé.org

Detalhando cada uma das camadas fica ainda mais simples:

- Fabric: referente a estrutura física do Grid e é correspondente à camada de recursos e a camada de rede, na listagem anterior.
- Recursos e protocolos de conectividade: gerência as transações específicas do grid, bem como a conectividade deste com os recursos disponíveis. Um pilar fundamental deste conceito é a segurança.
- Serviços coletivos: fornece informações sobre o estado e a estrutura do grid, bem como gerência o acesso aos recursos disponíveis. Listando os serviços (e dados) executados nesta camada é

Por fim, é interessante notar que ambas as definições das camadas de um grid tratam a camada de aplicação como a camada mais alta e visível pelo usuário.

1.6. Utilização

Sistemas em Grid são utilizados para os mais diversos fins. Entre alguns dos problemas que os sistemas em Grid são capazes de resolver encontram-se os problemas de “grande desafio” (Grand Challenge). Um problema de grande desafio é um tipo específico de problema para o qual não existe solução conhecida e que se caracteriza, basicamente, por pelo menos uma das seguintes características:

- requer avanços significativos na capacidade requerida para resolve-lo.
- deve ter uma solução, idealmente deve fornecer uma maneira plausível de quantificar o progresso em relação à solução final do problema.
- a solução do problema tem um impacto econômico ou social significativo.

Entre os problemas de grande desafio, pode-se citar: enovelamento de proteínas (processo pelo qual uma proteína assume sua forma funcional), modelagens financeira e climática, simulações complexas em geral, etc.

Para tornar ainda mais simples o entendimento, podemos separar os problemas que os clusters podem resolver, da seguinte maneira:

- de acordo com o seu grau de paralelismo, isto é, de acordo com a quantidade de pequenas operações que podem ocorrer simultaneamente.
- de acordo com a sua granularidade, ou seja, de acordo com a inter-dependência do resultado de pequenas operações em curso, variáveis armazenadas no sistema. Este ainda se subdivide em “fine-grained” e “coarse-grained” (também conhecidos como “embarassingly parallel”). Grosseiramente falando, problemas puramente “fine-grained” se saem melhor em grandes, monolíticos supercomputadores ao passo que “coarse-grained” se saem melhor em grids.

Problemas destacados como “fine-grained” são também identificados como sendo do tipo “high performance computing” e os “coarse-grained” como “high throughput computing”. É interessante notar, entretanto, que muitos dos problemas existentes são uma mistura de ambos os tipos.

Em geral, estes problemas caracterizam-se por serem grandes demais para serem processados por um único cluster ou supercomputador. De certa forma, neste tipo de problema é possível obter uma vazão de dados e processamento muito maior através da utilização de um sistema em Grid do que utilizando um supercomputador de capacidade de processamento semelhante.

1.7. Grid x Cluster

Uma das controvérsias existentes em torno dos sistemas em Grid, deve-se ao fato destes serem comumente confundidos com clusters.

Porém, antes de definir as diferenças entre um cluster e um Grid é preciso conhecer a definição de um cluster e como este trabalha. A definição de Buya[1] para cluster é a que segue: “A cluster is made up of multiple interconnected nodes that co-operatively work together as a single unified resource. Unlike Grids, clusters resources are owned by a single organization and they are managed by a centralized resource management and scheduling system. That means that all users of a cluster have to go through a centralized system that manages allocation of resources to application jobs.” (trad.: um conjunto de múltiplos nós interconectados que trabalham cooperativamente juntos como um único recurso. Ao contrário dos Grids, os recursos de um clusters são pertencentes a uma única organização e eles são gerenciamentos por um recurso de gerenciamento e escalonamento centralizado. Isto significa que os usuários de um cluster tem que passar por um sistema centralizado que gerencia a alocação de recursos para os trabalhos das aplicações).

Através dessa definição de Buya fica bastante claro afirmar quais são os pontos chaves nos quais os clusters se diferem dos grids

- os clusters são fisicamente centralizados, isto é, o membros (nós) de um cluster encontram-se dispersos sobre uma mesma área física (um prédio, sala, datacenter, etc).
- os recursos (poder de processamento, memória, etc) de um cluster são administrados pela organização responsável pelo cluster. Em um Grid, a administração deste recurso cabe a cada um dos responsáveis pelos nós do Grid.

Além disso, segundo a definição de cluster utilizada por Buya, bem como o que foi estudado anteriormente, é fácil compreender alguns dos outros aspectos que definem a diferença entre ambos:

- os grids, devido a sua estrutura descentralizada, têm uma disposição de recursos computacionais muito mais heterogênea do que um cluster. Ou seja, a variação do poder de processamento, memória, disco, etc dos membros de um grid é muito maior do que aquela encontrada nos membros de um cluster (aonde o que se deseja geralmente é o contrário, caso contrário, poderia configurar-se como um gargalo).
- os membros (nós) de um grid não precisam estar permanentemente inter-conectados.
- clusters tendem a serem utilizados para solução de problemas lineares, ao passo que Grids devem ser utilizados para sistemas capazes de serem processados em paralelo.

Por fim, é importante ressaltar que é possível criar grids utilizando clusters como membros, entretanto o contrário não é possível.

1.8. Grid x Peer To Peer (P2P)

Dada sua natureza também descentralizada, não seria de se admirar que alguém se perguntasse quais as diferenças e semelhanças entre um grid e uma rede p2p.

Segundo Ledlie[3] em seu artigo “Scooped Again”, tanto sistemas peer to peer quanto sistemas em Grid compartilham de um conjunto de problemas em comum.

É importante notar, porém, que peer to peer diz respeito à infra-estrutura e design de uma rede. Portanto é possível afirmar que peer to peer diz respeito à infra-estrura de acesso, compartilhamento e busca de informações, ao passo que Grid diz respeito ao acesso e compartilhamento de recursos computacionais.

1.9. Grid x Supercomputadores

Supercomputador é um termo, de uso amplo, utilizado para definir recursos computacionais de altíssimo desempenho. Idependente da sua estrutura física e lógica, este termo é utilizado para definir recursos computacionais como: clusters, grids, etc.

Supercomutadores, do ponto de vista de uma estrutura física e computacional única, também podem fazer parte de um Grid.

1.10. Exemplos de Grids

Finalizando a introdução sobre Grids, é interessante exemplificar a sua utilização atual, de modo a poder avaliar um pouco da sua aplicação prática. Como exemplo de sistemas em Grid, podem ser citados:

- Seti@home: Search for Extra-Terrestrial Intelligence (Busca por Inteligência Extra-Terrestre). Este projeto é utilizado para analisar os dados recebidos pelo rádio-telescópio Arecibo, localizado em Arecibo – Porto Rico.
- LHC@home: utilizado para melhorar o acelerador de partículas LHC (Large Hadron* Collider – Grande Colisor de Hadrons). Obs.: Hadron é uma partícula sub-atômica de grande força nuclear.
- Climaprection.net: visa melhorar a previsão do clima à longo prazo.
- Predictor@home: utilizado para prever a estrutura de uma proteína a partir de uma seqüência protéica.

2. Global Grid Forum

O Global Grid Forum (GGF) é uma entidade que reúne usuários, empresas e desenvolvedores de Grids no mundo todo. Entre alguns dos membros do GGF é importante citar: Nasa, IBM, Intel, Microsoft, Oracle, Cisco, Novartis, etc. É interessante notar que entre os membros do GCF não encontram-se apenas empresas de tecnologia, como é o caso da já citada Novartis, que atua na industria farmaceutica.

O principal trabalho do GGF consiste em definir padrões, políticas e boas práticas relacionadas ao desenvolvimento de grids, criando também, uma comunidade internacional de troca de idéias, experiências e requerimentos relacionados à computação em grid.

O GGF é a entidade que produz o GGF Document Series, uma série de documentos que definem os padrões de funcionamento (por exemplo, autenticação, comunicação, transmissão de dados, etc) de um grid. É baseado nestes padrões que a Globus Alliance desenvolveu o Globus Toolkit.

3. Globus Alliance

A Globus Alliance é uma comunidade internacional cujo objetivo é pesquisar e desenvolver as tecnologias fundamentais para o desenvolvimento e implantação de um grid. Entre os participantes membros to “core team” do projeto encontram-se: o Laboratório Nacional Argonne da Universidade de Chicago, Universidade de Endinburgo (EPCC), Centro Nacional de Aplicações de Supercomputadores (NCSA), Laboratório de Computação de Alta Performance da Universidade do Norte de Illinois, Instituto Real de Tecnologia da Suécia, Corporação Univa e o Instituto de Informações da Universidade do Sul da California. Além destes

participantes, inúmeras outras universidades ao redor do mundo contribuem para o projeto.

Membros da Globus Alliance participam em uma grande variedade de projetos de computação em Grid, nas mais diversas áreas: astronomia, química, engenharia civil, meteorologia, geologia, medicina, etc.

A grande contribuição da Globus Alliance para a pesquisa e desenvolvimento de grids chama-se Globus Toolkit, que é um conjunto de bibliotecas e programas utilizados no desenvolvimento e implantação de Grids.

4. Globus Toolkit

O Globus Toolkit, também conhecido como GT4, onde 4 refere-se a sua versão, é um dos mais famosos e usados conjuntos de ferramentas para desenvolvimento e implementação de Grids. O GT é um projeto de código-fonte aberto/livre iniciado por volta de 1998 pela Globus Alliance e é inteiramente desenvolvido implementando padrões abertos.

O GT permite que se compartilhe, com segurança através de uma rede, banco de dados, poder de processamento e muito mais.

O GT4 provê ferramentas e meios para gerenciamento de recursos, segurança, infra-estrutura, portabilidade, tolerância a falhas e muito mais, ao mesmo tempo respeitando a singularidade de cada entidade que possa vir a utiliza-lo.

Muitas entidades, empresas e universidades ao redor do mudo são usuários do Globus Toolkit, entre elas podemos citar:

- Centro de Terremotos do Sul da California, o qual utiliza o GT para visualização de dados simulação de de terremotos. Estas simulações cobrem uma vasta área e utilizam-se de gráficos de alta-resolução, podendo cada simulação, chegar a 40TeraBytes de dados.

- O CERN utiliza o GT nos seus grids de simulação de colisões de partículas.
- Os cientistas do Earth Grid System (EGS) utilizam o GT para prover acesso, armazenar e processar dados de pesquisas climáticas da Terra.

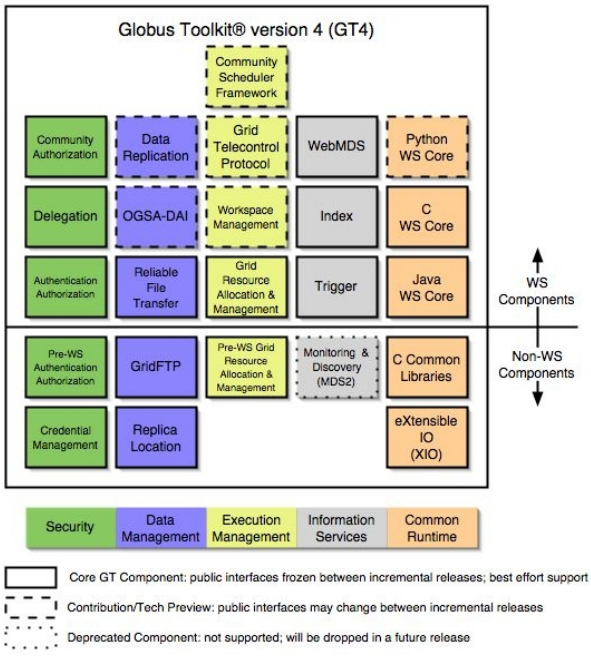
Assim como acontece em um grande número de projetos OpenSource, o GT conta com uma vasta gama de canais de fornecimento de suporte, contando com chats, listas de discussões e workshops para treinamento e suporte.

4.1. Arquitetura

O GT4, como mencionado anteriormente, é um conjunto de ferramentas (aplicativos e bibliotecas) para desenvolvimento de sistemas distribuídos.

De modo geral, o GT4 é estruturalmente dividido em:

- camada de segurança
- camada de gerenciamento de dados
- camada de gerenciamento de execução
- informações de serviços
- common runtime (sistemas de execução comuns)



Por fim, ainda divide-se entre as componentes Web Services e pré Web Services.

4.1.1.Common Runtime

O Common Runtime é um conjunto de bibliotecas e ferramentas cujos objetivos são prover um conjunto de serviços Web e pré-web independente de plataforma, permitir a construção e o desenvolvimento desses serviços em múltiplas camadas e aumentar a funcionalidade nas camadas mais baixas da pilha de serviços.

O Common Runtime implementa uma enorme quantidade de protocolos padrão web, entre os quais é importante citar: XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), WSDSL (Web Services Description Language), TLS (Transport Layer Security), X.509, HTTP (HyperText Transfer Protocol), SMTP (Simple Mail Transfer Protocol) e muitos outros.

4.1.1.1.Bibliotecas Comuns C

O conjunto de bibliotecas comuns C fornece uma camada de abstração entre a aplicação e o sistema operacional, a biblioteca C existente e estruturas de dados utilizados por todo o toolkit.

É importante lembrar, também, que estas bibliotecas comuns funcionam em diversos sistemas operacionais, tornando mais fácil o porte de um aplicativo para outra plataformas. As plataformas atualmente suportadas são: Linux, FreeBSD, HP/UX, AIX, Tru64 Unix, Windows, Solaris.

Por fim, é interessante ressaltar a importância deste componente, uma vez que este conjunto de bibliotecas é usado em praticamente todo o toolkit, devido as camadas de abstração serem desenvolvidas na linguagem C.

4.1.1.2.C Web Services Core

O C Web Services Core (C WS Core) provê uma vasta gama de ferramentas para implementação de serviços e clientes web utilizando a linguagem de programação C, a partir da qual é possível fazer ligações (“bindings”) para diversas outras linguagens, como Fortran, C++, Pascal e muitas outras.

O C WS Core é incluí:

- Um container para serviços
- Uma interface de programação de aplicações (API, Application Programming Interface, em inglês) plugável para serviços.
- Uma API para gerenciamento de recursos
- Uma API para gerenciamento de clientes notificadores
- Bindings (ligadores) geradores de WSDL para C.
- Serviços de apoio e suporte a segurança.

O WS Core conta também com ferramentas de linha de comando para suporte as aplicações desenvolvidas nessa camada, como uma ferramenta para hosperdar servicos web C e outra para gerar “esqueletos” em C.

Uma das grandes características do C WS core, e é bastante dificil não atentar para este fato, é a sua extensibilidade. Ele é tão extensível, que em sua documentação o autor demonstra como utilizar os bindings WDSL C na criação de um serviço de blog (um assunto completamente fora do escopo da computação distribuída).

Por fim, o C WS Core suporta os seguintes padrões:

- HTTP
- SOAP
- XML Schema
- WSDL
- WS Security
- WS Addressing
- WS Resource Framework
- WS Notification

4.1.1.3.Java WS Core

Assim como o C WS Core, o Java WS Core fornece funcionalidades de serviços Web, entretanto este, ao contrário do primeiro, fornece para a linguagem Java.

O Java WS Core é divido em duas partes o serviço e o recurso. O serviço é responsável por executar a lógica de negócios no recurso, sendo que o recurso representa um estado gerenciável.

O Java WS Core implementa os padrões WSRF e WSN (Web Services Notification).

4.1.1.4.Globus XIO – eXtensible Input Output

O Globus XIO é a biblioteca de entrada e saída do Globus, e através da qual conexões são abertas e fechadas.

Implementada na linguagem C e definida através da utilização de funções de callback, ela torna possível a utilização de um modelo de programação assíncrono baseado em eventos.

Esta biblioteca permite, entre outras coisas , que usuários definam “drivers” através dos quais os dados enviados pelos usuários poderão passar, servindo de propósito específico à aplicação.

4.1.2.Camada de Segurança

Esta é a camada responsável pela autorização e autenticação dentro do Globus Toolkit. Ela é subdividada em duas partes menores: CAS (Central Authentication Service – Serviço Central de Autenticação) e Delegation Service (Serviço de Delegação).

4.1.2.1.CAS

É o responsável por gerenciar as políticas de acesso em uma VO (Virtual Organization – Organização Virtual).

As políticas de acesso são armazenadas em um banco de dados, que é acessador através de uma interface de administração.

4.1.2.2.Serviço de Delegação

Através da utilização de uma estrutura de chaves público-privadas, permite a delegação de credenciais utilizadas para acessar recursos no/do grid.

4.1.3.Camada de Gerenciamento de Dados

A Camada de Gerenciamento de Dados é a responsável por armazenar, transferir e gerenciar os dados distribuídos. Ele é constituído de 3 ferramentas principais: GridFTP, Reliable File Transfer Service (RFT service, serviço de transferência confiável) e Replica Location Service (RLS, serviço de localização de réplicas).

4.1.3.1.GridFTP

O GridFTP é um das ferramentas/biblioteca utilizadas pelo Globus Toolkit para implementar transferência de arquivos de maneira eficiente.

O GridFTP suporta não apenas o protocolo FTP tradicional, mas também algumas extensões cuja finalidade é deixa-lo mais seguro. Além disso permite a inclusão de plugins cujo objetivo é aumentar a funcionalidade e a tolerância a falhas do conjunto.

Por fim, é importante notar que o GridFTP é um protocolo padrão, conforme definido pelo Global GridForum em conjunto com uma série de RFCs (Requests For Comments) da IETF (Internet Engineering Task Force).

Embora o GridFTP com toda sua simplicidade seja uma ferramenta poderosa, em muitos casos ele não é um serviço eficiente para transferência de dados, além de contar com algumas pontos negativos que podem ser de extrema importância em alguns casos. Como falhas do GridFTP pode-se citar o fato de ele mantém um socket permanentemente aberto com o servidor durante a transferência de dados o que acaba se tornando um empecilho durante longas transferências de dados. Outro detalhe a ser lembrado a respeito do GridFTP é que este não é um WSP.

4.1.3.2.RFT Public Interface

Visando atender as deficiências existentes no GridFTP, o RFT é um serviço que implementa o padrão WSP, além de ser concordante com o WSRF.

No caso do Globus Toolkit, o RFT é disponibilizado através de classes em Java, entretanto, segundo a documentação oficial disponível no site, este ainda encontra-se em processo de melhoramento.

Adicionalmente é interessante notar que o serviço RFT e implementado tendo o GridFTP como base.

4.1.3.3.RLS Public Interface

O RLS é o serviço responsável por localizar as réplicas de dados nos dispositivos de armazenamento físico que compõem o grid.

O RLS atua como um registro múltiplo, mantendo informações sobre os arquivos em diversos servidores, aumentando a disponibilidade dos dados e diminuindo os pontos de falha.

Para evitar confusões com entre os nomes de arquivos, o RLS utiliza um esquema de nomes de arquivos lógicos e nomes de arquivos físicos, sendo que o primeiro é um identificador único para um arquivo e o segundo identifica a localização deste no sistema de armazenamento.

4.1.4.Camada de Gerenciamento de Execução

O Globus Toolkit fornece ferramentas para submeter, monitorar e cancelar “jobs” (trabalhos) em sistemas grids que utilizem o GT. Jobs são trabalhos computacionais cuja execução pode gerar entrada/saída. No Globus Toolkit o conjunto de ferramentas que permite este controle é conhecido como Gram.

Em muitos casos é desejado pelos usuários de um sistema de monitorar e verificar a execução de dos dados relacionados a um job. Atender a esses requisitos, no Globus Toolkit, é função da camada de gerenciamento de execução.

4.1.5.Informações de Serviços

É a camada responsável por, entre outras coisas monitoração e descobrimento de sistemas e recursos disponíveis no Grid. No Globus Toolkit é composto pelos WS MDS e Pre WS MDS.

4.1.5.1.WS MDS

O WS MDS (Monitoring and Discovery System) é o sistema cuja finalidade é permitir e facilitar ao usuários a descoberta e monitoração de recursos disponíveis em uma organização virtual. No Globus Toolkit é composto pelos seguintes componentes:

- Agregator Framework: utilizado para construir serviços de coleta e agregação de dados.
- Information Providers: uma fonte de dados utilizada pelo serviço agregador.
- Index Service: um serviço de agregação e indexação de dados.
- WebMDS: um front-end web para o Index Service.

5. BOINC

O BOINC (Berkley Open Infraestructure for Network Computing – Infraestrutura Aberta Berkeley para Computação em Rede), é uma infraestrutura para desenvolvimento e implementação de sistemas distribuídos. Um dos grandes diferenciais do BOINC é o fato de que diversos projetos podem, nativamente, compartilhar os mesmos recursos.

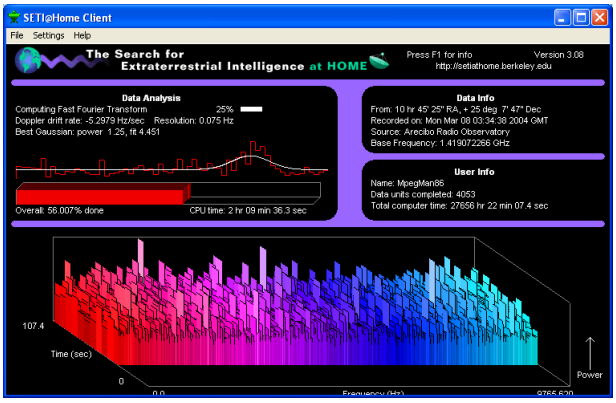
Um dos objetivos do BOINC é oferecer funcionalidades poderosas, porém de fácil utilização e desenvolvimento. Entre as funcionalidades disponibilizadas pelo BOINC pode-se citar:

- Framework de desenvolvimento flexível: aplicações nas mais diversas linguagens (C, C++, Fortran) podem usar o BOINC com pouco ou nenhum esforço.
- Segurança: garante a autenticidade e a confiabilidade dos dados através do uso de criptografia e a autenticações por chaves públicas.
- Suporte a múltiplos servidores e

tolerância a falhas: suporta escalonadores separados e conta com um algorítimo inteligente para evitar que clientes sobrecarreguem o servidor após um down-time.

- Código-fonte disponível: distribuído sobre a LGPL (Lesser General Public License), permite, entre outras coisas, que seu código fonte seja distribuído em conjunto com software proprietário/fechado.
- Múltiplataforma: o BOINC funciona em Linux, Windows, MacOS e outros sistemas operacionais.

O BOINC é utilizado por uma série de projetos, muitos deles já citados neste artigo. Provavelmente o mais famoso dos usuários do BOINC é o Seti@home, projeto que visa identificar a existência de vida extraterrestre.



6. Conclusão

A tecnologia de computação em Grid é um exemplo magnifico de como o a extensa criatividade e inteligência dos cientistas resulta em soluções práticas e, principalmente, acessíveis, capazes de resolver mesmo os mais complexos problemas.

Por fim, o apanhado geral sobre o Globus Toolkit, oferece uma visão do design e implementação de um sistema de grid (mesmo que este seja apenas um framework), tornando mais claro o entendimento do sistema como um todo.

7. Anexos

7.1. Autorização para uso de imagens

Algumas das imagens utilizadas neste artigo são de propriedade do CERN e foram gentilmente cedidas, sem qualquer custo, para uso neste artigo através dos seguintes emails:

De: Otavio R. Piske <angusyoung@gmail.com>
Para: Rosy Mondardini <gridcafe.info@cern.ch>

Hi, I'm writing an article as a graduation exam for my postgraduation course and I want permission to use the following graphic in my material:
http://gridcafe.web.cern.ch/gridcafe/gridatwork/images/gridlayer.gif
There won't be any comercial usage of this document. It'll only be available to other students in my university.

Thanks In Advance
Otavio R. Piske

De: Rosy Mondardini <gridcafe.info@cern.ch>
Para: Otavio R. Piske <angusyoung@gmail.com>

Dear Otavio, please feel free to go ahead and use the images that you need, just remember, please, to add somewhere close to them "Courtesy of www.gridcafe.org"

Best Regards
Rosy

Este artigo conta, ainda com imagens fornecidas pela Globus Alliance e Globus Toolkit, conforme autorizadas pelos emails a seguir:

De: Otavio R. Piske <angusyoung@gmail.com>
Para: Rosy Mondardini <liming@mcs.anl.gov>
Hi, I'm writing an article as a graduation exam for my postgraduation
course and I want permission to use the following graphics in my article:
http://www.globus.org/toolkit/docs/4.0/GT4figure.jpg
http://www-unix.globus.org/alliance/impact/scec-dfm-pair.jpg
http://www-unix.globus.org/alliance/impact/leadimpact.jpg
http://www-unix.globus.org/alliance/impact/scec-terashake-x.gif

There won't be any comercial usage of this document. It'll only be available to other students in my university (http://www.unicenp.edu.br/ingles/).

Thanks In Advance
--
Otavio R. Piske – AngusYoung

De: Rosy Mondardini <liming@mcs.anl.gov>
Para: Otavio R. Piske <angusyoung@gmail.com>

You may use the first image without any concern at all.

The last three images were obtained from our science partners and we (the University of Chicago) do not have authority to give you permission to use them.

For the two SCEC images (second and fourth), please contact Marcus
Thiebaux <thiebaux@ISI.EDU>.

I do not have a current contact for the CERN lead ion image, but the
CERN publicity office
(http://info.web.cern.ch/Press/ContactUs.html)
would be a place to start. (We obtained permission to use it at least
two years ago.)

Os outros gráficos utilizados nesses textos encontravam-se disponíveis para uso sem custo ou necessidade de requisitar autorização.

8. Referências Bibliográficas

1) Dr. Rajkumar Buyya: http://www.gridcomputing.com/gridfaq.html http://gridbus.org/press/EA03/EAInterview.pdf
2) Ian Foster: http://www.gridtoday.com/02/0722/100136.html
3) Jonathan Ledlie, Jeff Shneidman, Margo Seltzer, John Huth – Scooped Again. http://iptps03.cs.berkeley.edu/
4) Seti@home: http://setiweb.ssl.berkeley.edu/
5) LHC@home: http://athome.web.cern.ch
6) ClimaPrediction.net: http://climateprediction.net/
7) Predictor@home: http://predictor.scripps.edu/
8) Grid Café: http://gridcafe.web.cern.ch
9) CERN: http://www.cern.ch
10) Globus Toolkit: http://www.globus.org
11) Globus Toolkit Documentation: http://www.globus.org/toolkit/docs/4.0/public_interfaces.html
12) BOINC: http://boinc.berkeley.edu/