Introdução aos sistemas de computação em grade: Globus, Legion, Globe e Condor

Mac 449/5775 – Sistemas Operacionais Distribuídos

Andrei Goldchleger
andgold@ime.usp.br

Motivação

- Grande necessidade de computação: para que?
- Simulações
  - Financeiras, Mercadológicas
  - Climáticas, ambientais
- Estudos variados
  - Previsão do Tempo
- Aplicações na Indústria
  - Manipulação de multimídia
  - Prospecção de Petróleo

Máquinas paralelas e supercomputadores

- Muitas vezes multiprocessados
- Processadores integrados por hardware proprietário, dependendo de redes de comunicação proprietárias
- Caras (custo de dezenas de milhares de dólares, as vezes até milhões)
- Acesso restrito a:
  - Instituições de pesquisa de 1o nível
  - Grandes empresas
- Possuir uma máquina desse tipo, devido ao custo, normalmente só se justifica quando há uma grande utilização da mesma

Aglomerados (*Clusters*) de computadores tipo PC (1/2)

- Tipicamente, uma potencia par de PCs (4,8,16...) conectados por *fast ethernet*
- Implementação de cluster: Beowulf (1994) é o exemplo mais famoso
- União de diversos pacotes independentes
  - MPI/PVM (Bibliotecas de programação paralela)
  - Patches para o kernel do Linux (Exemplo: endereçamento de processo global)

Aglomerados (*Clusters*) de computadores tipo PC (2/2)

- Vantagens dos Aglomerados:
  - Baixo custo em relação à máquinas paralelas tradicionais
  - Pode ser montado com hardware comum
  - Extensível facilmente
- Desvantagens dos aglomerados:
  - Normalmente dedicados (Inclusive fazem essa suposição)
  - Ainda caros para instituições com recursos financeiros limitados
  - Ociosidade
  - Ocupam espaço, esquentam, fazem barulho...

Atualmente (1990+, ganhando força em 1995+)

- Integrar recursos em escala mundial, extrapolando os limites locais
- Computação em Grade (GRID Computing) (Ian Foster, 1998)
- Permitir o compartilhamento de recursos distribuídos, estejam eles onde estiverem
- Operar sobre os diversos SOs, arquiteturas e redes de comunicação, abstraindo assim tais detalhes e facilitando o desenvolvimento de aplicações para a grade
- A maioria é Middleware
- Oferecem alguns serviços presentes nos SOs mas que atuam de maneira global.

Globus

- Iniciado em 1997
- www.globus.org
- Argonne National Labs, U. of Chicago, U. of Southern California
- Ian Foster, Carl Kesselman, Steve Tuecke
- É o sistema de computação em grade mais popular
- Código Aberto
- Grande atenção por parte da mídia, inclusive atraindo a atenção de empresas (exemplo: IBM)
- Atualmente na versão 3.x, o que aproximadamente representa a terceira geração do sistema
- Escrito em C

Globus: Características Fundamentais

- Arquiteturado como toolkit, permitindo a criação de aplicações que usufruam dos serviços da grade de maneira incremental
- Exemplo:
  - Inicialmente, pode-se usar Globus apenas para agendar a execução em múltiplas máquinas
  - Posteriormente, pode-se adicionar uma biblioteca para detecção e correção de falhas
  - Finalmente, pode-se utilizar os serviços Globus de distribuição de arquivos
- Diversos serviços, como escalonamento, segurança, informações e distribuição de dados

GRAM (*Globus Resource Allocation Manager*)

- Um dos serviços de escalonamento do Globus
- Cada GRAM cuida de uma ou mais máquinas, tipicamente um conjunto local
- Serviço global é composto da união de vários GRAM
- RSL(*Resource Specification Language*) permite especificar os recursos necessários a uma aplicação
- GRAM não possui suporte para co-alocação de recursos
  - Exemplo: Aplicação precisa de 10 nós, sistema aloca 8 e não consegue os outros 2. Execução é abortada.

GARA (*Globus Architecture for Reservation and Allocation*)

- Também é um módulo de escalonamento de aplicações
- Serviço mais avançado que GRAM
- Permite reservas de recursos
- Possui mecanismos para garantir QoS das aplicações

MDS (*Metacomputing Directory Service*)

- Mantém informações sobre as diversas máquinas que compõem a grade
  - hardware, SO, memória, latência de rede
- Implementado utilizando LDAP (*Lightweight Directory Access Protocol*)
  - Permite armazenar e consultar dados
- Serviço de informação em escala global é a reunião de diversos servidores MDS

GSI (*Globus Security Infrastructure*)

- Lida apenas com autenticação
- Mapeia as credenciais de acesso para mecanismos locais de autenticação
- Usuário autentica uma vez no começo da sessão, e recebe uma credencial válida por toda uma sessão de uso

CoG Kits (*Commodity Grid Kits*)

- Iniciativa para levar o poder da grade à aplicações do usuário comum
- Esforços para fornecer APIs do Globus em diferentes linguagens:
  - Java
  - Python
  - Web Services
  - CORBA
  - JSP
  - PERL
  - Matlab

OGSA (*Open Grid Services Architecture*)

- Presente na 3a geração do toolkit
- Tentativa de juntar Computação em Grade e *Web Services*
- Os serviços do Globus possuiriam assim uma casca implementada em Web Services, facilitando assim a adoção de serviços da grade por usuários de Web Services
- Talvez o assunto mais comentado do momento em termos de Computação em Grade

Legion

- 1993-2001?
- U. da Virginia
- http://legion.virginia.edu/
- Andrew Grimshaw
- Aparentemente a pesquisa no projeto terminou (último artigo é de 2001)
- Não disponível para download: parece ser comercializado por uma empresa, a Avaki

Legion: Princípios de Arquitetura

- Autonomia para diferentes domínios
- Núcleo extensível
- Arquitetura escalável
- Ambiente de computação homogêneo
- Espaço de nomes único e persistente
- Aproveitamento de recursos heterogêneos
- Suporte à múltiplas linguagens e interoperabilidade
- Tolerância à falhas
- Executar com poucos privilégios

Legion: Objetos e Classes

- Arquitetura totalmente orientada à objetos
  - Tudo no sistema são objetos
- Classes: possuem responsabilidades de sistema
  - Criação de objetos
  - Ativação/Desativação
  - Agendamento da execução
- Objetos comunicam-se por chamadas de métodos assíncronas
- Interfaces definidas por um tipo de IDL
- Legion possui protocolos e ORB próprios

Legion: Identificadores de Objetos

- Hierarquia composta por 3 níveis
  - Nomes de contexto: Cadeias de caracteres com significado. Análogo a nomes de domínio registrados no DNS
  - LOID (*Legion Object Identifier*): Identificador único de um objeto. Análogo a IORs em CORBA
  - LOA (*Legion Object Address*): Especifica uma maneira de se comunicar com um objeto (Ex: TCP -> IP/porta)

Legion: Objetos Centrais

- Implementam funcionalidades básicas necessárias a todos os objetos
- Existem implementações de referência, mas podem ser reimplementados pelos programadores
- Exemplos de objetos centrais
  - Objetos de contexto: mapeiam nomes de contexto a LOIDs
  - Agentes de associação: guardam pares (LOID,LOA). Podem implementar políticas diferenciadas
  - Objeto hospedeiro: Representa um processador na grade
  - Objeto cofre: representa um objeto Legion qualquer seriado em disco
  - Objeto de implementação: representa um programa a ser executado

Legion: Segurança

- Associada a possibilidade de se chamar os diferentes métodos dos objetos
- Método "MayI": Todo objeto possui, e indica os privilégios
- Usuário recebe certificado listando seus direitos
- MayI intercepta chamadas e verifica se o usuário que as fez possui tais direitos
- Objeto possui par de de chaves criptográficas
  - Pública: embutida no LOID, permite comunicações criptografadas
  - Privada: permite que objetos assinem suas mensagens

Legion: Programação paralela e Linguagens em geral

- Possui suporte a bibliotecas paralelas:
  - MPI
  - PVM
- Linguagens suportadas:
  - MPL (Extensão de C++)
  - BFS (Basic Fortran Support)
  - Java
- Aplicações legadas podem ser encapsuladas em objetos

Globe

- Iniciado em1995
- Vrije Universiteit, Holanda
- http://www.cs.vu.nl/globe/
- Andrew Tanenbaum, Maarten van Steen
- Objetivo principal: construção de sistemas distribuídos escaláveis em redes de grande área
- Motivação: cada sistema distribuído implementa o seu protocolo de comunicação, políticas de *caching e* replicação. Globe pretende ser um arcabouço provedor de tais funcionalidades

Globe: Objetos Distribuídos Compartilhados

- Todos os elementos do sistema são objetos
- Um único objeto pode estar fragmentado em diferentes máquinas
- Acesso ao sistema se dá através de métodos
- Alguns exemplos de funcionalidades que podem ser implementadas:
  - devolver o conteúdo de uma página da teia (como um servidor HTTP)
  - devolver o conteúdo ou parte de um arquivo
  - receber como argumento uma mensagem de email (como um servidor SMTP)
  - realizar uma operação em um banco de dados de escala mundial
  - qualquer coisa que se faz em um sistema distribuído mas sempre usando o mesmo modelo OO

Globe: Sub-objetos

- Componentes de um objeto Globe
- Cada objeto possui uma certa quantidade de sub-objetos
- Cada sub-objeto é responsável por uma determinado aspecto do funcionamento do objeto
- Exemplos de sub-objetos:
  - Semântica: contém os aspectos funcionais do objeto
  - Comunicação
  - Replicação: pode determinar uma política de replicação específica
  - Controle
- O modelo comportaria outros sub-objetos para a implementação de outros aspectos, como segurança, persistência...

Globe: Sub-objeto de Replicação

- Tem de lidar com a consistência das réplicas
- Podem implementar diferentes políticas de consistência
- Premissa básica: consistência pode ser determinada pela possibilidade de se chamar os métodos de um objeto
- Intercepta as chamadas, mantendo assim a consistência das réplicas de acordo com a política estabelecida

Globe: Sub-objeto de Controle

- Intercepta as chamadas aos sub-objetos de semântica
- Responsável por empacotar/desempacotar as chamadas que ocorrem entre ele e o sub-objeto de replicação
- Permite o acesso e modificação do estado do objeto de semântica

Globe: Sub-objeto de Comunicação

- Responsável pela comunicação entre as partes de um objeto
- Pode utilizar diversos tipos de comunicação
  - Conectada X Não-conectada
  - Confiável X Não-confiável
  - Ponto-a-ponto X Multicast
- Encapsula a comunicação atrás de uma interface padronizada
- Independente dos sub-objetos de semântica

Condor

- Iniciado em 1988
- http://www.cs.wisc.edu/condor/
- U. of Wisconsin at Madison
- Miron Livny
- Derivado de um projeto ainda mais antigo, o Remote Unix (1986)
- É o pioneiro dos sistemas de computação em grade
- Licença de código nebulosa, mas parece que está sendo aberto sobre licença liberal
- Objetivo principal de Condor: utilizar o potencial ocioso dos computadores

Condor: *High Throughput Computing*

- Condor introduziu conceito de HTC
- HTC = Necessidade de processamento por longos períodos: dias, semanas, meses…
- Contraste com HPC: FLOPS
- Este é o grande atrativo: utilizar tempo ocioso para cumprir tarefas de HTC

Condor: Organização Básica

- Sistemas Condor se organizam em aglomerados
- Um aglomerado tipicamente compreende as máquinas em uma rede local
- 3 tipos de nós em uma aglomerado:
  - Nó fornecedor de recursos
  - Nó consumidor de recursos
  - Nó gerenciador do aglomerado

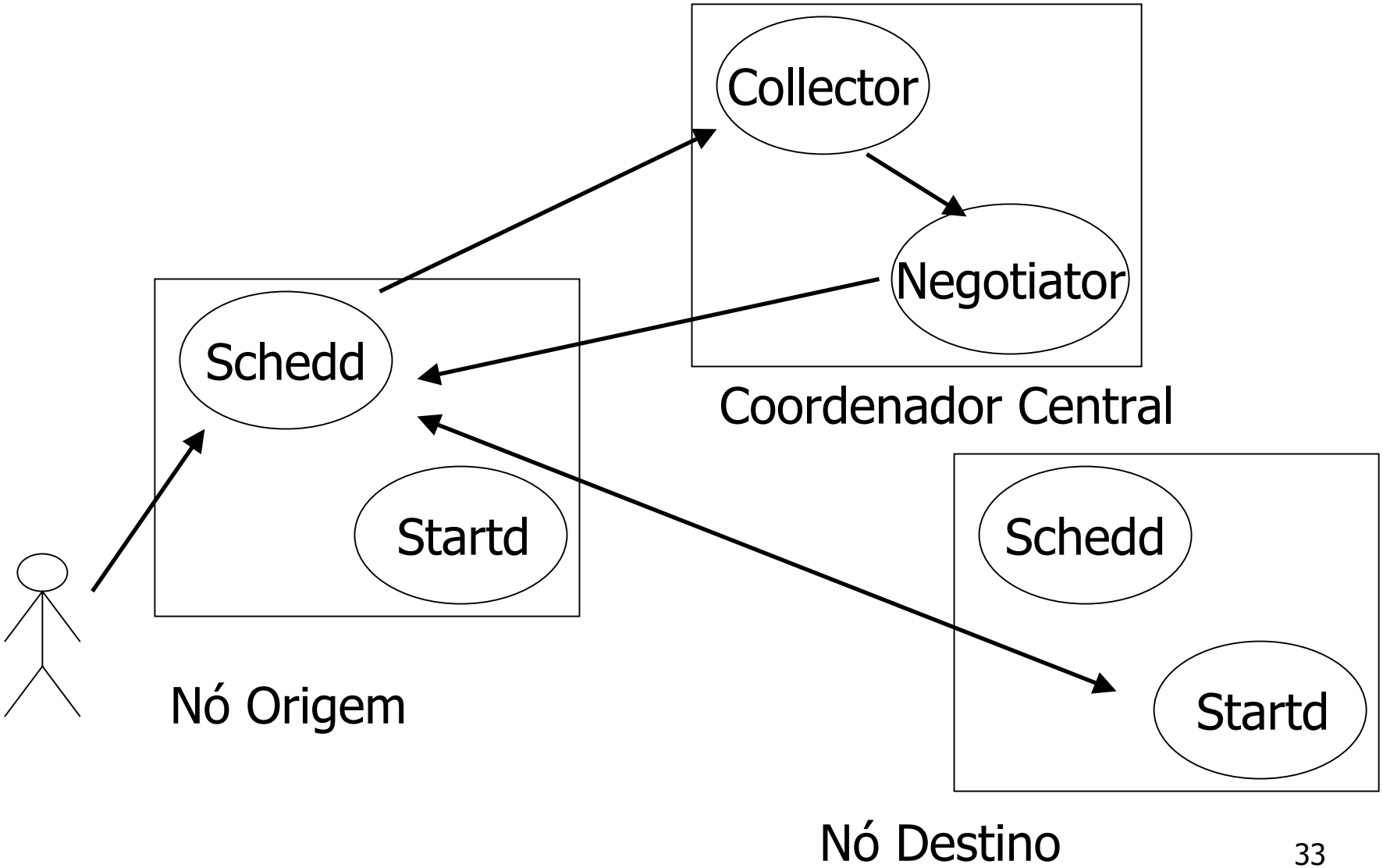
Condor: Módulos

- Schedd
  - Mantém fila de tarefas submetidas pelo dono da máquina
  - Obtém recursos para executar tarefas
- Startd
  - Anuncia recursos do nó ao Coordenador Central
  - Aplica a política de uso do proprietário do nó
- Collector
  - Recebe informações e requisições de todas as máquinas do aglomerado
- Negotiator
  - Com base nos dados de Collector, realiza o emparelhamento entre oferta e procura

Disseminação de Informações: ClassAd

- Estrutura de Dados que representa informações
- Não possuem esquema fixo
- Entradas são do tipo (atributo, valor)
- Podem representar:
  - Requisições feitas por um Schedd:
    - "Execute este programa em máquinas com 128Mb de memória, ao menos”
  - Ofertas de recursos feitas por um Startd:
    - "Esta maquina é um Pentium III 500, com 512Mb de memória,....”

Sequencia de Operação



Processo Sombra

- Após a obtenção do recurso remoto, inicia-se um processo sombra na máquina origem
- Funções:
  - Mascarar complexidade de processo remoto
  - Receber estado da execução (erros…)
  - Receber resultados
  - Permitir a execução de chamadas de sistema remotas

Chamada de Sistema Remota

- Utilizadas tipicamente para acesso remoto a arquivos
- Chamadas de sistema capturadas e enviadas para execução na máquina origem
- Problemas:
  - Ineficiente, especialmente se o programa utiliza muito
  - Nessessidade de religar o código com a biblioteca de suporte

Checkpointing

- Possibilidade de salvar estado de execução
- Mover aplicação quando máquina torna-se ocupada
- Salva todo estado da aplicação: pilha,registradores…
- Problemas:
  - Em aglomerados heterogêneos possibilidade de migração é reduzida
  - Não podem ser utilizados recursos como: Kernel-level threads, fork, pipes e memória compatilhada
  - Não funciona com aplicações paralelas
  - Depende de muitas informações do SO, como sinais pendentes, arquivos abertos e estado da CPU

Aplicações Paralelas

- Condor suporta aplicações escritas em MPI-CH e PVM
- Recursos de um *cluster* são marcados como dedicados, o que significa:
  - Nunca desalojam aplicações paralelas
  - Dão preferência a aplicações paralelas
- Não há *checkpointing* de aplicações paralelas
- Vantagens:
  - Melhor que aglomerado dedicado, pois evita particionamento de recursos
  - Máquinas do cluster podem executar aplicações seqüenciais na ausência de aplicações paralelas
- Desvantagens:
  - Alguns recursos permanecem semi-reservados
  - Inviável para estações de trabalho

Condor Multi-aglomerado

- Diversas formas de integração de aglomerados:
  - Gateway Flocking: Transparente
  - Direct Flocking: Usuários se registram em diversos aglomerados
- Condor e Globus
  - Condor-g: Schedd modificado que acessa grades globus via GRAM
  - Aglomerados condor *ad-hoc*
    - Utiliza-se Condor-g para submeter startds à grade globus
    - Startds contactam Collector/Negotiator especificado pelo usuário
    - Cria-se assim um aglomerado Condor em cima de parte de uma grade Globus