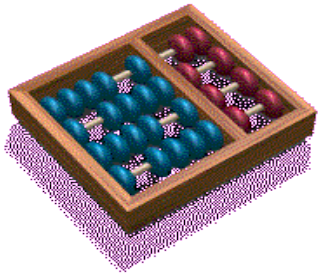
Universidade Estadual de Campinas
Instituto de Computação
Laboratório de Redes de Computadores - LRC



Middeware para execução de Tarefas Dependentes em Grades Computacionais

Edmundo R. M. Madeira
edmundo@ic.unicamp.br

Série de Seminários - IC

Campinas, 12 de Setembro de 2008

Middleware

Laboratório de Redes de Computadores

- Entre o Sistema Operacional e as Aplicações
- Suporte Transparente a Distribuição das Aplicações

Tarefas Dependentes

Laboratório de Redes de Computadores

- De Dados (tarefas seqüenciais e paralelas)
- modeladas como um grafo acíclico direcionado - DAG

Grade Computacional

Laboratório de Redes de Computadores

- sistema distribuído heterogêneo
- possui uma estrutura de gerenciamento e coordenação
- controla a execução de tarefas, a disponibilidade dos recursos e os serviços que fazem parte da infra-estrutura

Computação em Grade

Laboratório de Redes de Computadores

- compartilhamento coordenado de recursos por *organizações virtuais* multi-institucionais e dinâmicas
- organização virtual é um conjunto de indivíduos ou instituições que fazem o compartilhamento de recursos coordenado e controlado, com fornecedores e consumidores de recursos que definem claramente o que pode ser compartilhado e sob que condições esse compartilhamento ocorre.

Roteiro

Laboratório de Redes de Computadores

1. Middleware para execução de processos estruturados – Fábio Cicerre / Luiz Eduardo Buzato
2. Algoritmos de escalonamento para execução de tarefas dependentes – Luiz Fernando Bittencourt
3. Ambientes de Grade no contexto da Web 2.0 Colaborativa para E-Ciência – Carlos Roberto Senna

Laboratório de Redes
de Computadores

1. Middleware para execução de processos estruturados

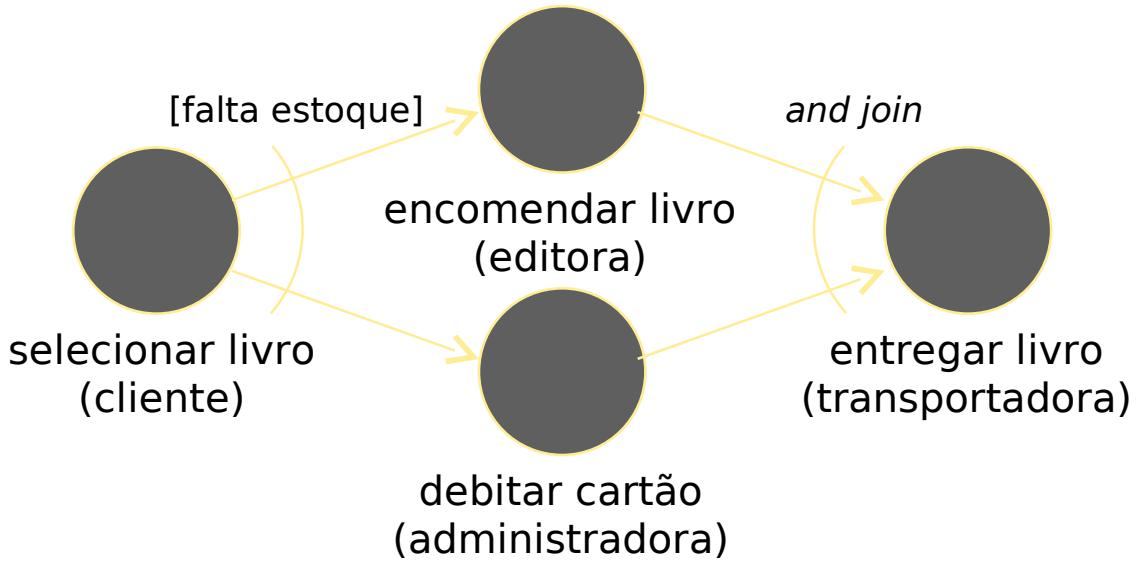
Introdução

Laboratório de Redes de Computadores

- **Grade Computacional**
  - Objetivos
    - Acesso transparente aos computadores, serviços e dados compartilhados entre várias organizações
    - Fornecer poder computacional não disponível em um único computador ou aglomerado
  - Funcionalidades
    - Autenticação, autorização e comunicação segura
    - Registro e monitoramento de recursos e serviços
    - Transferência de arquivos e dados
    - Acesso a recursos computacionais para execução de tarefas isoladas
  - Exemplos
    - Globus (GT4), Condor-G e Legion

Sistemas de Workflow

- **Especificação de workflows**
  - Atividades a serem executadas (o que)
  - Ordem de execução (quando)
  - Dados a serem utilizados (com o que)
  - Agentes responsáveis (quem)



Suporte a Processos em Grade

Laboratório de Redes de Computadores

- Funcionalidades
  - Execução e monitoramento de processos
    - Escalonamento, distribuição e coordenação de atividades
  - Adaptação dinâmica
  - Tolerância a falhas
    - Sistema
    - Processos e atividades
    - Dados
- Objetivos
  - Alto desempenho
  - Escalabilidade
  - Adaptabilidade
  - Confiabilidade e disponibilidade

Sistema Xavantes

Laboratório de Redes de Computadores

- Objetivo
  - Sanar algumas deficiências dos sistemas existentes e tornar o uso da grade computacional mais efetivo
- Composição
  - Linguagem estruturada
  - *Middleware* para execução de processos
- Contribuições
  - Linguagem de especificação simples e expressiva
    - Controladores e objetos compartilhados
  - Execução hierárquica sobre vários grupos
  - Escalonamento adaptativo com alocação e prioridades
  - Tolerância a falhas com recuperação

Trabalhos Relacionados

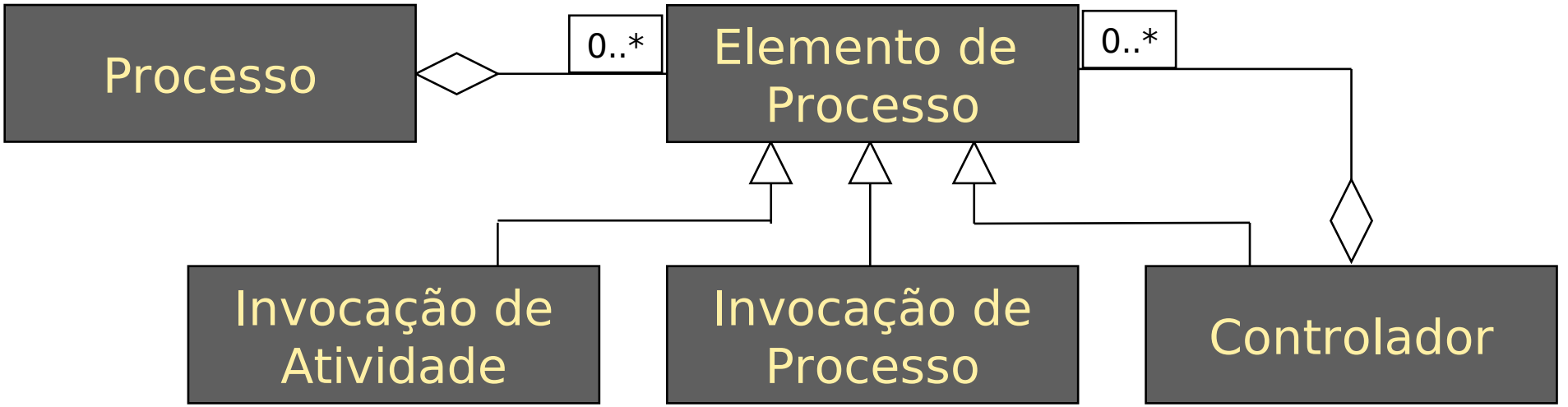
Aspecto / Sistema	Especificação	Coordenação da Execução	Escalonamento	Tolerância a Falhas
DagMan e GridAnt	regras, abstrata	no cliente	local, dinâmico	retentativa
UNICORE	DAG, concreta para dados	no servidor	local, determinado pelo usuário	*middleware*
Pegasus	DAG, abstrata para dados	distribuída	global, dinâmico	DagMan
Symphony e WebFlow	DAG, abstrata	no servidor	local, dinâmico	retentativa
SwinDew	DAG, abstrata	*peer-to-peer*	local, dinâmico	retentativa
Triana	componentes ligados, abstrata	*peer-to-peer*	local, dinâmico	*middleware*
Opera-G	DAG, abstrata	no servidor	local, dinâmico	retentativa, recuperação
GridFlow	DAG, abstrata	dois níveis	global, estático por simulação	recurso alternativo
ICENI	componentes ligados, abstrata	centralizada	global, dinâmico com alocação	*middleware*
Xavantes	estruturada, abstrata	hierárquica	global, dinâmico com alocação e adaptativo	tratamento, recuperação, dados

Linguagem Estruturada de Processos

Laboratório de Redes de Computadores

- Denominação
  - *Xavantes Structured Process Language* (XSPL)
- Objetivos
  - Simplificar especificação e monitoramento de processos
    - Linguagem derivada de Java
  - Permitir construir processos complexos por meio da composição de controladores
    - Controladores paralelos com objetos compartilhados
    - Tratamento de exceções
  - Explicitar dependências e composição do processo para a execução hierárquica
  - Permitir a manipulação dinâmica dos dados e da estrutura do processo

Modelo de Especificação



- Elementos de Processo
  - Processo: compõe e coordena as invocações de atividades e processos, por meio de controladores, para resolver um problema definido
  - Controlador: define a ordem de execução e o fluxo de dados entre elementos de processo
  - Atividade: realiza uma tarefa básica do processo

Atividade

Laboratório de Redes de Computadores

- Utilização
  - Computação intensiva
  - Obtenção e transformação de dados
  - Invocação de serviços remotos
- Especificação
  - Implementada em Java
  - Compiladas para otimização da execução
- Exemplo
  - Multiplicação de matrizes
  - Renderização de gráficos 3D
  - Cruzamento de dados

Controlador

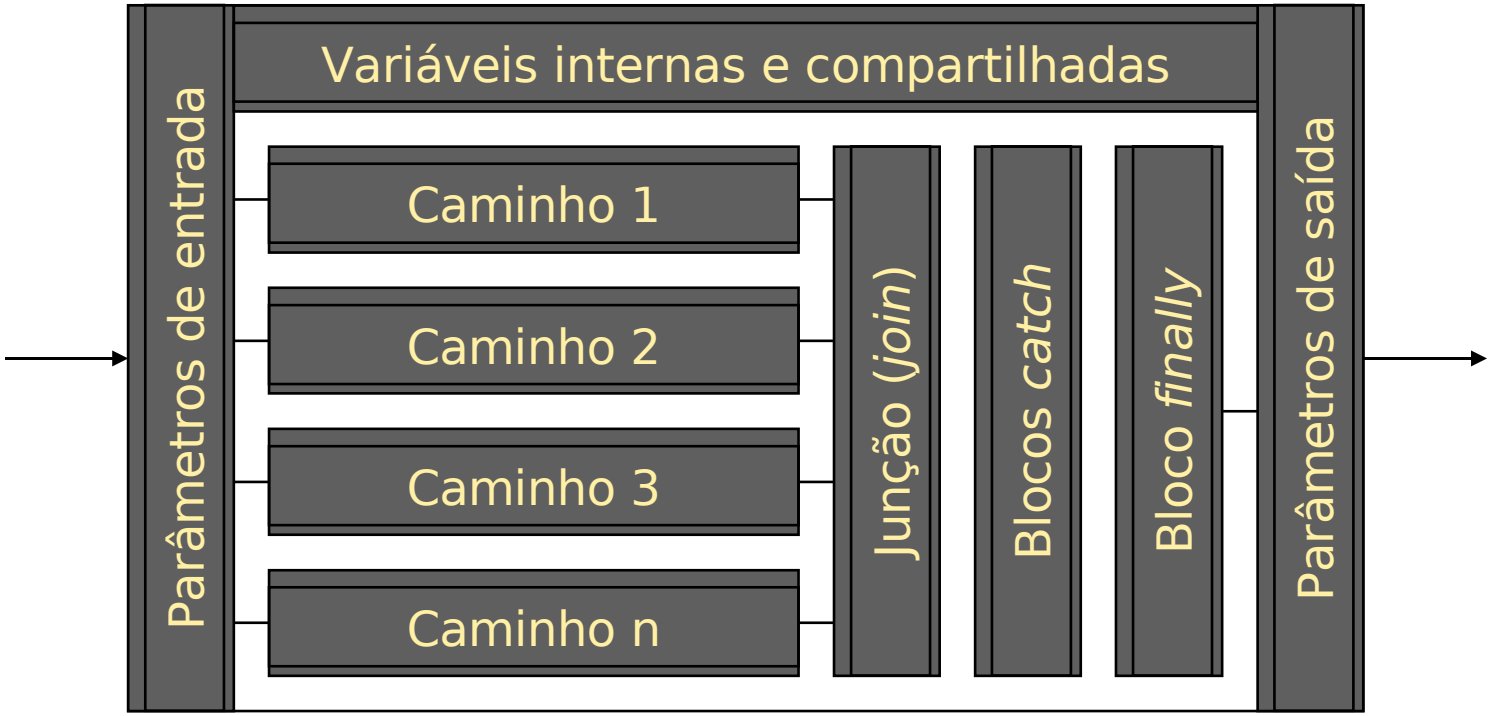
Laboratório de Redes de Computadores

- Utilização
  - Determinação da ordem de execução dos elementos de processo (invocações de atividades e de processos e controladores)
- Características
  - Estruturado e síncrono
- Classificação

Tipo / Paralelismo	Simples	Condicional	Iterativo	Iterativo-Condicional
Seqüencial	block	switch	for(each)	while
Paralelo	par	parswitch	parfor(each)	parwhile

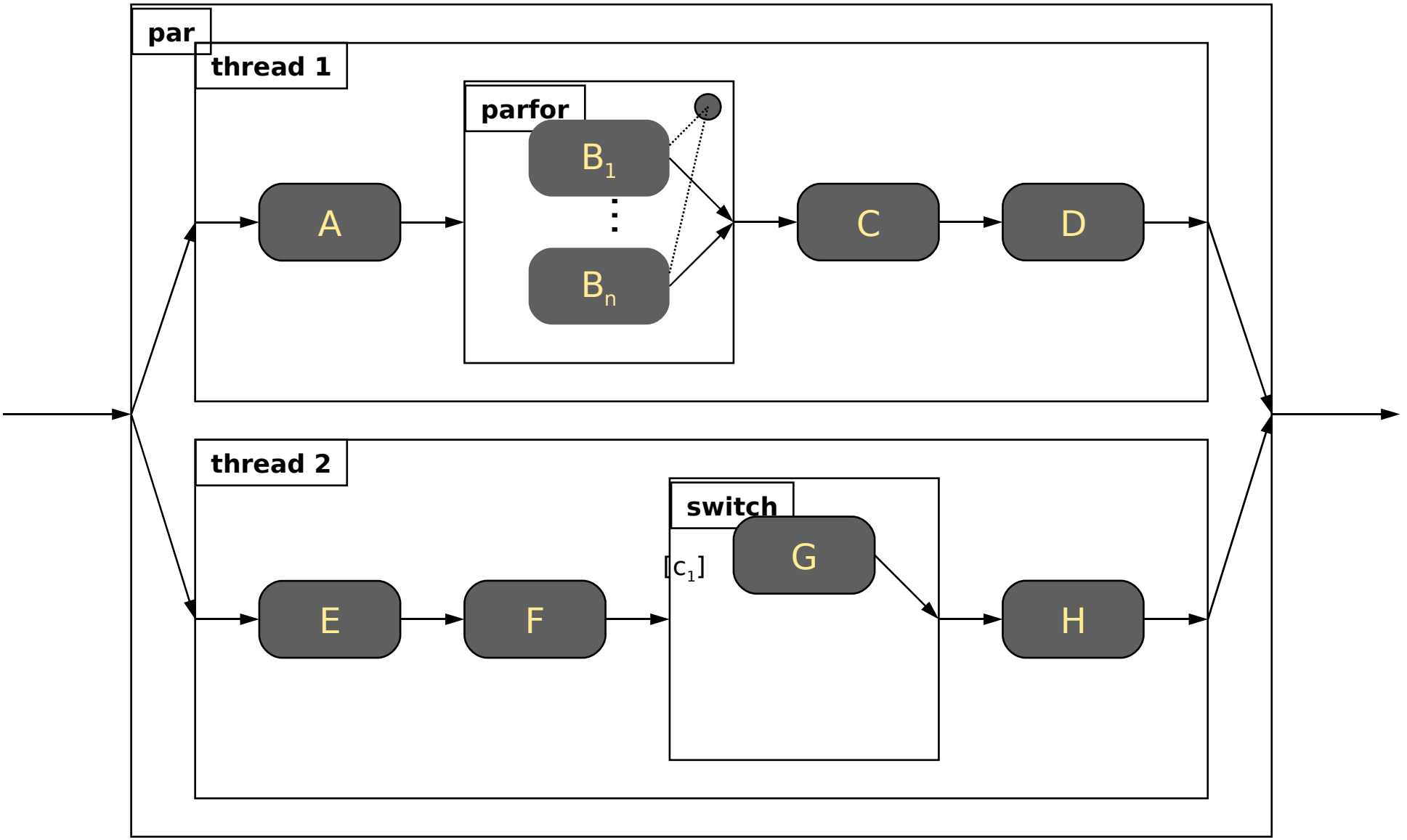
Controlador Paralelo (PAR)

Laboratório de Redes de Computadores



Fluxo de Controle e Dados

Laboratório de Redes
de Computadores



Linguagem – Outras Características

Laboratório de Redes de Computadores

- Controle fino
  - *Scripts* Java (junção flexível, expressões, dados)
  - Reflexão computacional
- Tolerância a falhas
  - Retentativas e replicação (atividade)
  - Recuperação
  - Tratamento de exceções
- Especificação de recursos para atividades
  - Existência de arquivos e fontes de dados
  - Quantidade de memória e poder de processamento
- Estimativa de uso de recursos
  - Quantidade de dados trocados com elemento superior
  - Esforço de processamento (atividade)

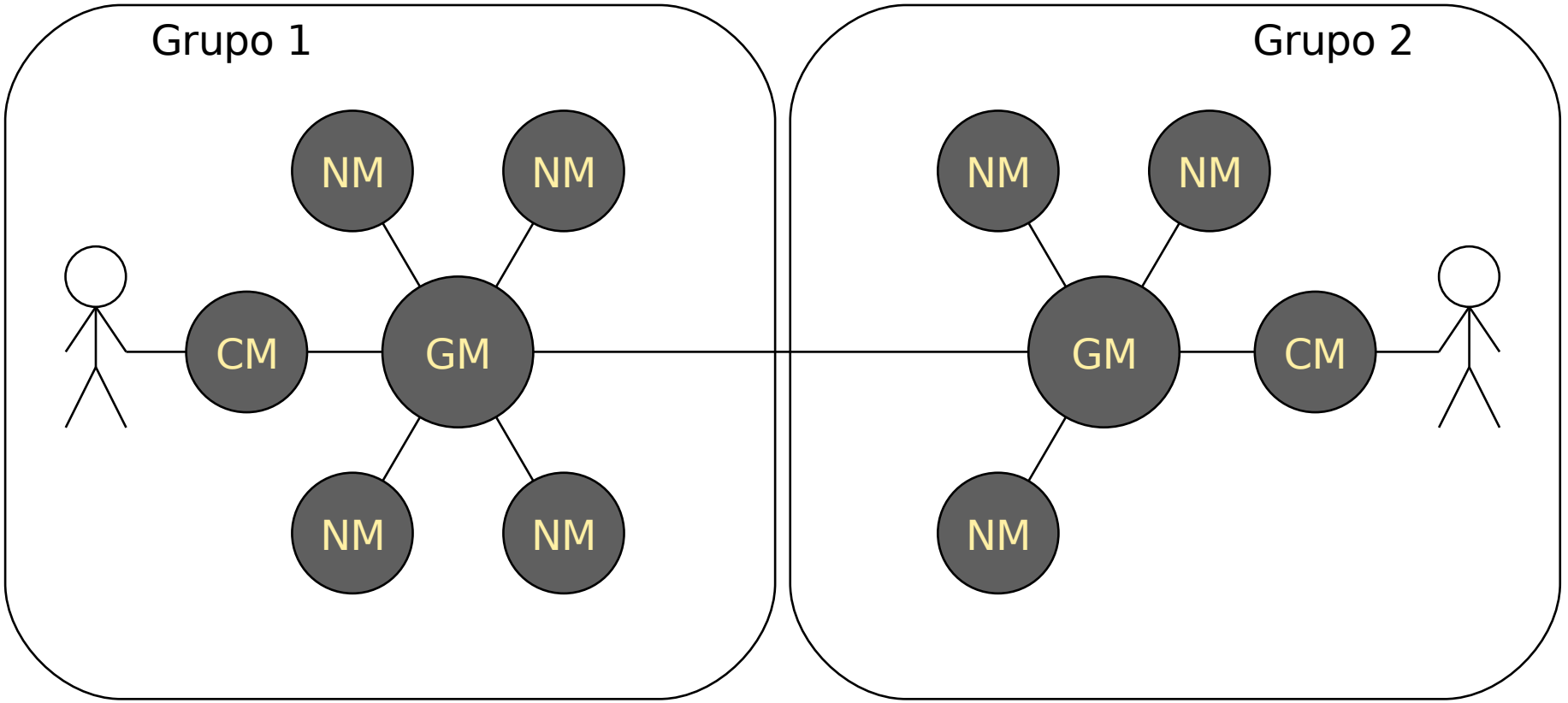
Middleware Xavantes

Laboratório de Redes de Computadores

- Objetivos
  - Monitorar a disponibilidade dos recursos
  - Prover acesso aos recursos
  - Executar processos estruturados na grade
    - hierarquicamente
    - em um ou mais grupos

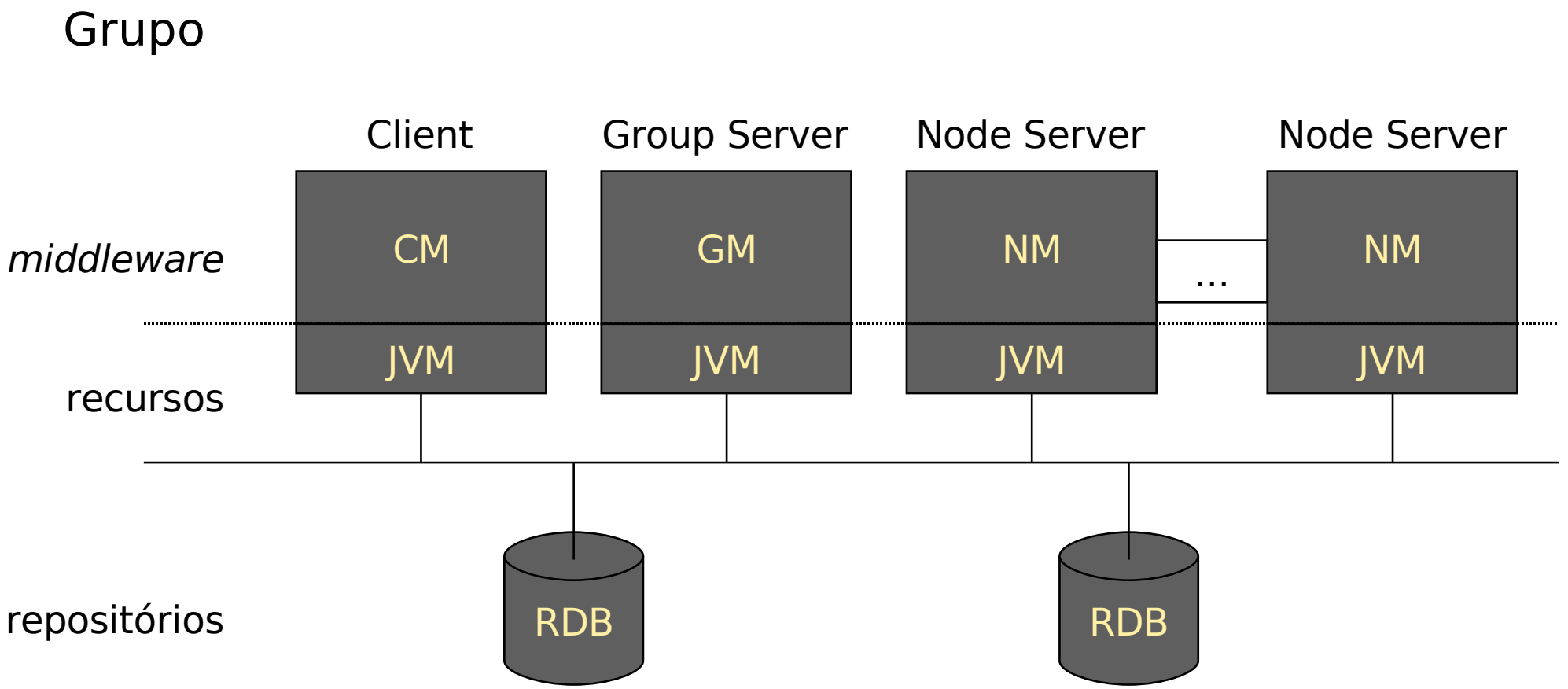
Distribuição do *Middleware*

Laboratório de Redes de Computadores



Arquitetura do *Middleware*

Laboratório de Redes
de Computadores



Serviços do *Middleware*

Laboratório de Redes de Computadores

- Gerenciador de Grupo (GM)
  - Registro de usuários
  - Monitoramento da disponibilidade dos recursos no grupo local e grupos externos
  - Alocação de recursos
  - Controle da execução de processos e atividades
- Gerenciador de Nó (NM)
  - Execução de processos e controladores, escalonando e coordenando a execução de elementos-filho
  - Execução de atividades
- Gerenciador de Cliente (CM)
  - Acesso ao recursos da grade
  - Controle da execução de processos de um usuário

Execução Hierárquica de Processos

Laboratório de Redes de Computadores

- Características
  - Execução hierárquica de processos, de acordo com a sua estrutura
    - Distribuição otimizada de controladores aos nós
      - Considera poder de processamento e de comunicação
    - Cada controlador controla o escalonamento e a coordenação de seus elementos-filho
  - Escalonamento e coordenação paralelos de elementos de processo
  - Escalonamento adaptativo baseado em prioridades
  - Recuperação distribuída (por controlador)
  - Comunicação localizada (*cache* distribuído)
- Potenciais conseqüências
  - Desempenho, escalabilidade, confiabilidade e adaptabilidade na execução de aplicações

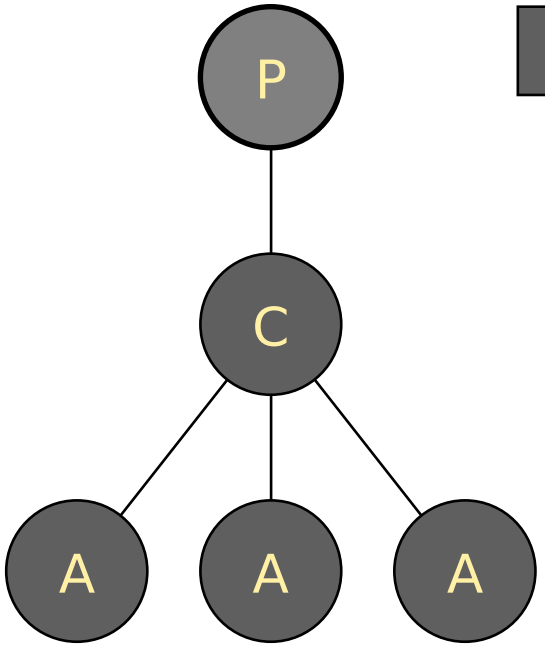
Etapas da Execução de Processos

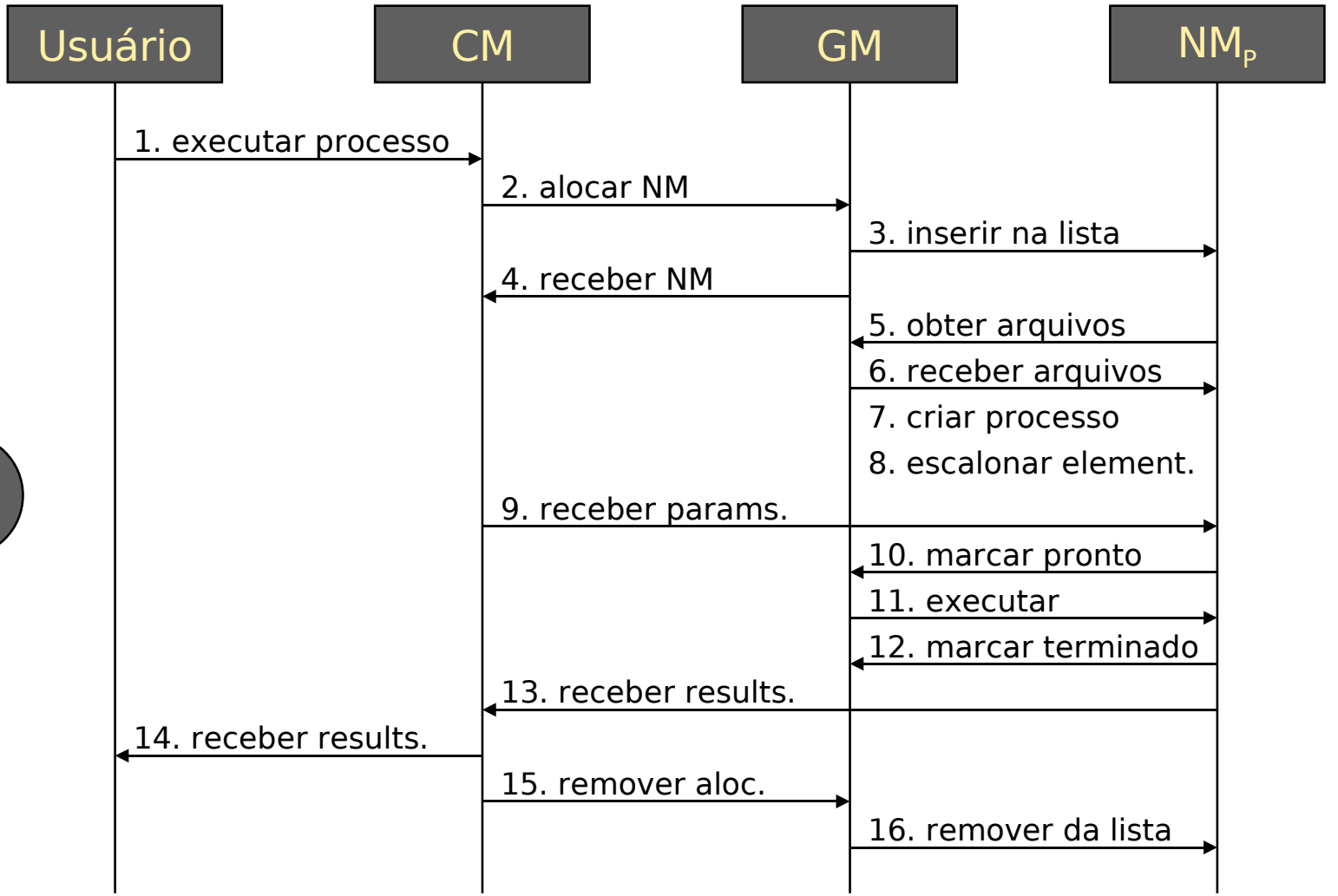
Laboratório de Redes de Computadores

- Escalonamento
  - Atribuição de prioridades de execução
  - Escolha de elemento a alocar (dep. de execução)
  - Requisição de alocação de servidor nó para elemento
- Coordenação da execução
  - Envio de parâmetros iniciais do elemento para servidor nó (no momento da execução)
  - Recebimento dos resultados do servidor nó
  - Remoção da alocação do elemento

Exemplo de Execução (Processo)

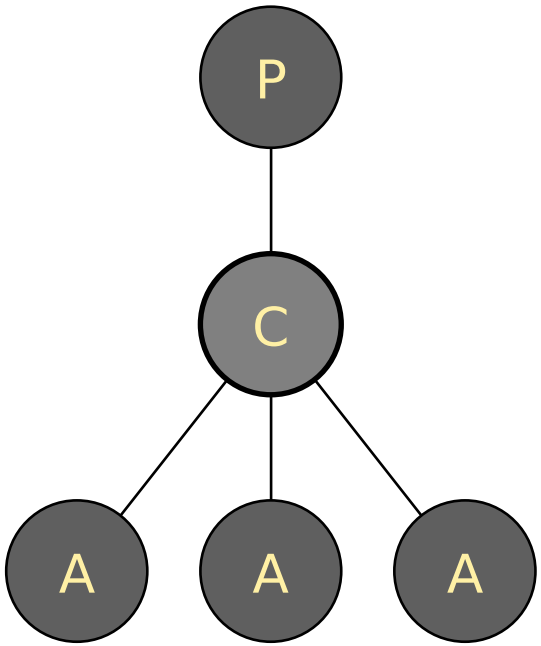
Laboratório de Redes de Computadores

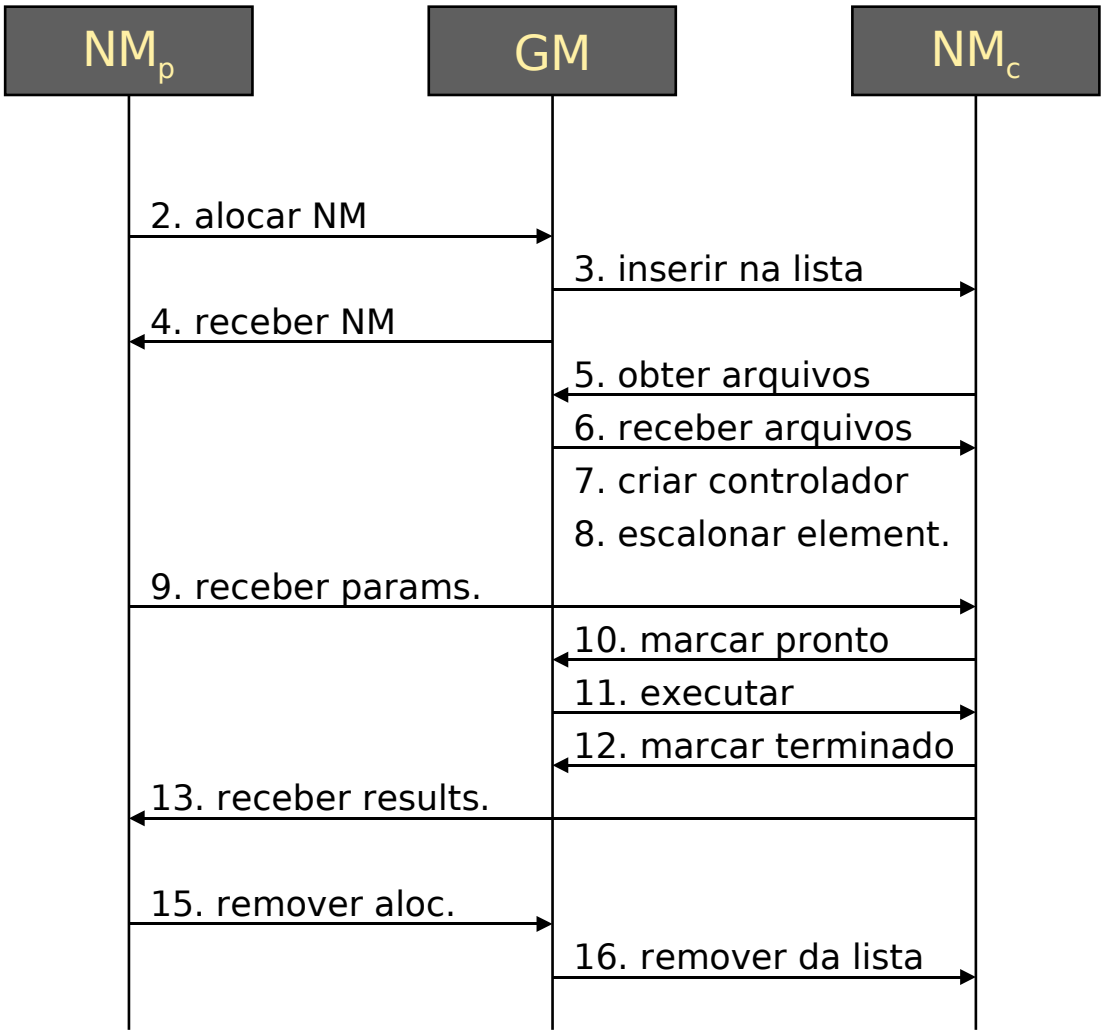




Exemplo de Execução (Controlador)

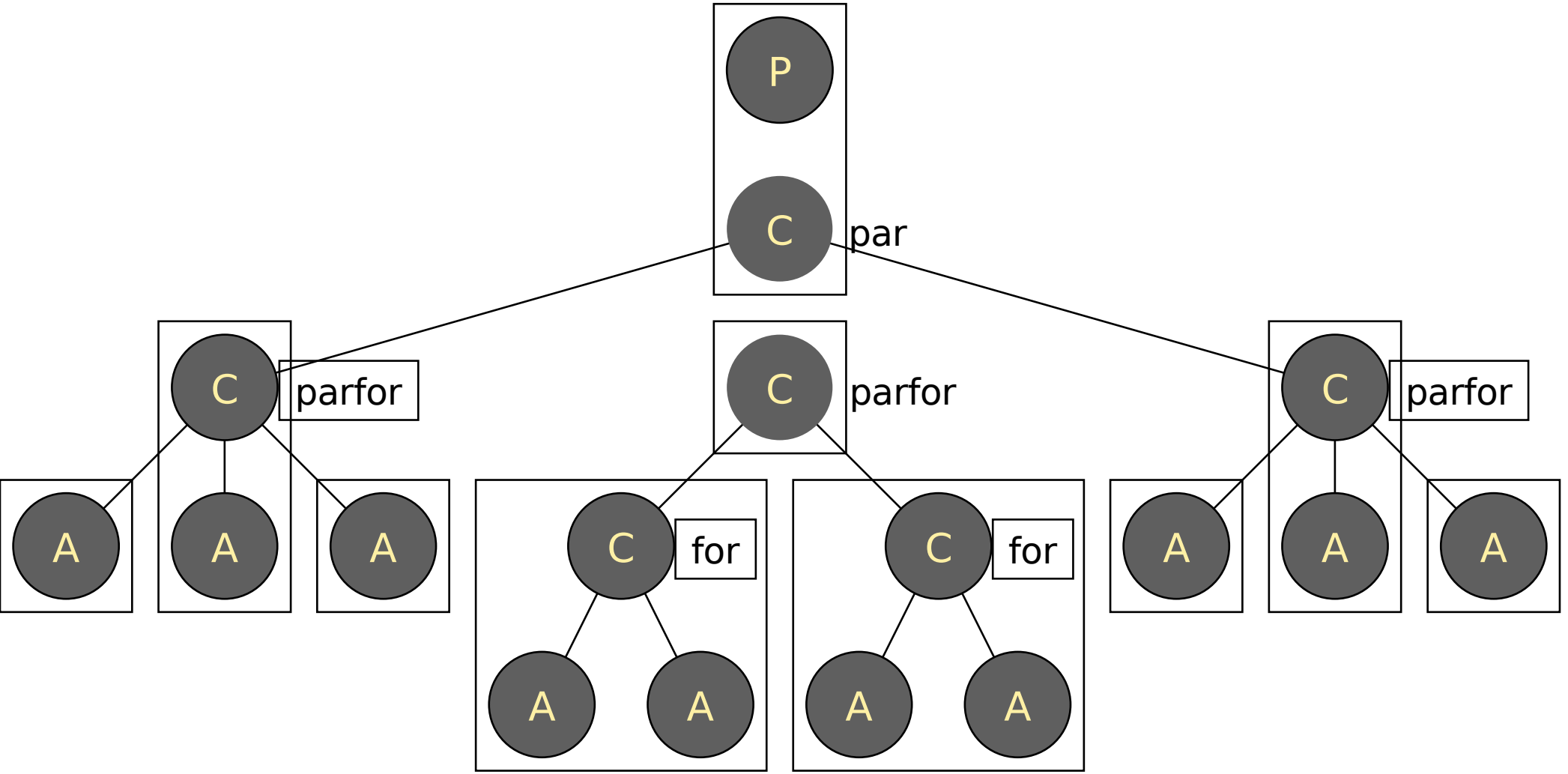
Laboratório de Redes de Computadores





Execução hierárquica

Laboratório de Redes de Computadores



Tolerância a Falhas

Laboratório de Redes de Computadores

- Nível do *middleware*
  - recuperação automática
- Nível de atividade
  - retentativa (mesma ou outra máquina)
  - recuperação automática (mesma ou outra máquina)
- Nível de processo
  - recuperação automática (mesma ou outra máquina)
  - tratamento de exceções geradas por falhas
  - modificação dinâmica do processo
- Objetos compartilhados
  - controle de concorrência (prevenção de erros)
  - restauração de dados

Implementação

Laboratório de Redes de Computadores

- Processo de desenvolvimento iterativo e evolutivo
  - Quatro protótipos foram implementados para embasar o projeto
  - *NetBeans IDE 5.5* - ambiente de desenvolvimento
  - Versão atual: Xavantes v0.2 (build 08/08/2007)
- Implementado com a linguagem *Java 6.0*
  - Pacote *Java Standard Edition 6.0* (*Java SE 6.0*)
  - *Remote Method Interface* (*RMI*) - comunicação
  - *Java Database Connectivity* (*JDBC*) - acesso BD
- *BeanShell 2.0* - execução de *scripts Java*
- *MySQL 5.0* - gerenciador de BD dos repositórios

Funcionalidades Implementadas

Laboratório de Redes de Computadores

Funcionalidade	Projeto	Implementação
Especificação de Processos	XML	Memória e BD
Execução de Processos	Hierárquica	Hierárquica
Distribuição	Vários Grupos	Máquinas de um grupo
Escalonamento	Distribuído, parcial	Distribuído, parcial
Alocação	Dinâmica por prioridades	Estática
Adaptação	Escalonamento parcial e realocação	Escalonamento parcial
Tolerância a falhas (atividades)	Retentativa, recuperação	Retentativa, recuperação
Tolerância a falhas (processo)	Recuperação, tratamento de exceções, modificação	Sem recuperação, tratamento de exceções
Tolerância a falhas (sistema)	Recuperação	Recuperação
Tolerância a falhas (dados)	Transações e referência persistente	Controle de concorrência e persistência (usuário)

Experimentos

Laboratório de Redes de Computadores

- Objetivos
  - Validar a execução de processos, adaptabilidade do escalonamento e tolerância a falhas
- Aplicação
  - Busca de uma certa quantidade de números primos em uma faixa definida (processo *FindPrimes*)
    - 100.000 números primos
    - na faixa de 2.000.000 a 10.000.000
    - utilizando 80 atividades paralelas
- Experimentos
  - Rodar a aplicação sobre grades com diferentes números de computadores e verificar o desempenho
  - Variar o número de computadores disponível e verificar a adaptação e tolerância a falhas na execução

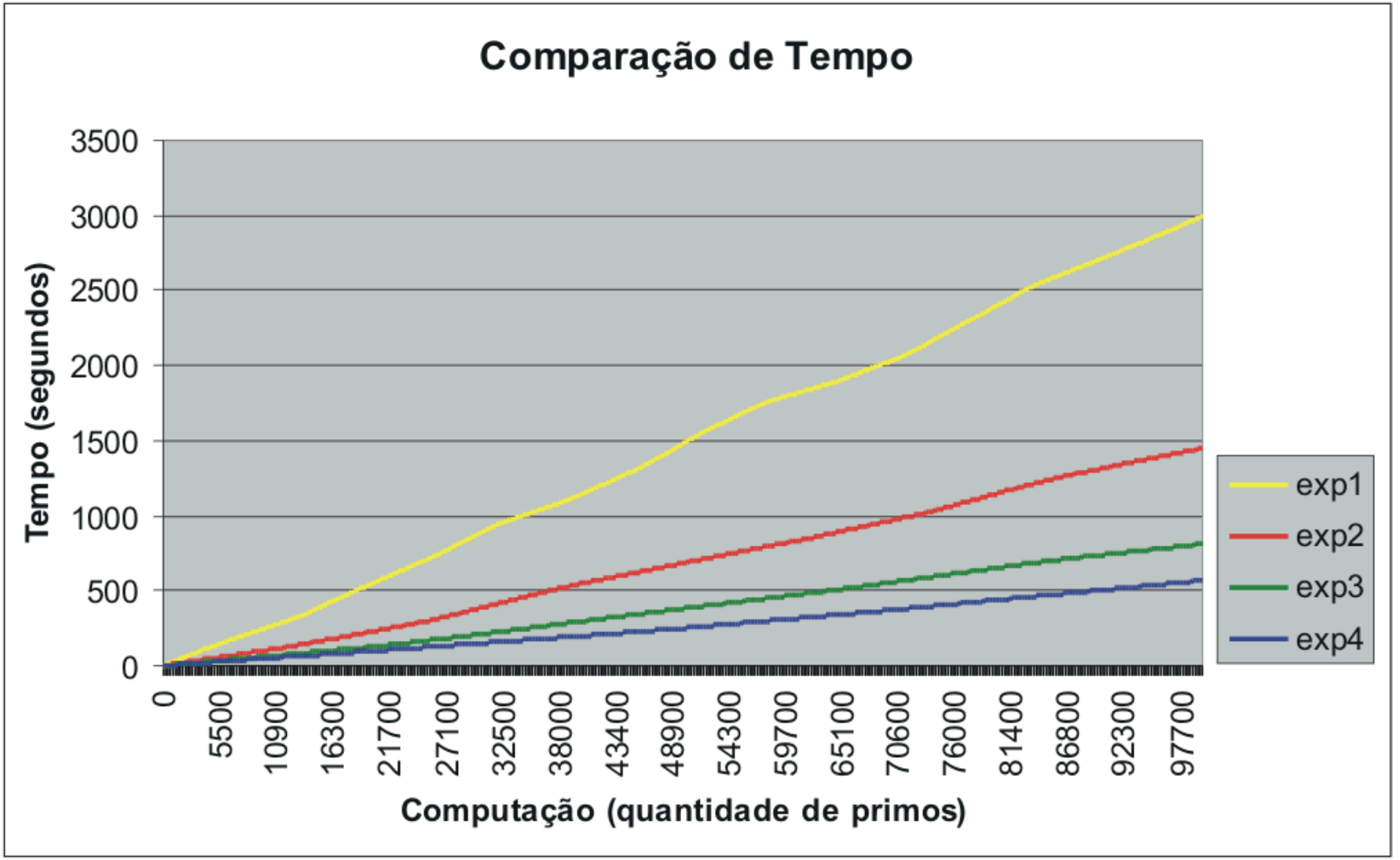
Número Fixo de Computadores

Laboratório de Redes de Computadores

- Cálculo de 100.000 números primos
- Foram realizados 4 experimentos
  - Exp.1: 1 computador - 5.858 MIPS
  - Exp.2: 2 computadores - 11.076 MIPS
  - Exp.3: 4 computadores - 18.604 MIPS
  - Exp.4: 8 computadores - 22.174 MIPS
- Resultados obtidos
  - Exp.1: 2.998 segundos (175,6 MIP / primo)
  - Exp.2: 1.448 segundos (160,4 MIP / primo)
  - Exp.3: 812 segundos (151,1 MIP / primo)
  - Exp.4: 569 segundos (126,2 MIP / primo)
- Avaliação
  - O *middleware* utiliza proporcionalmente todo o poder computacional disponível

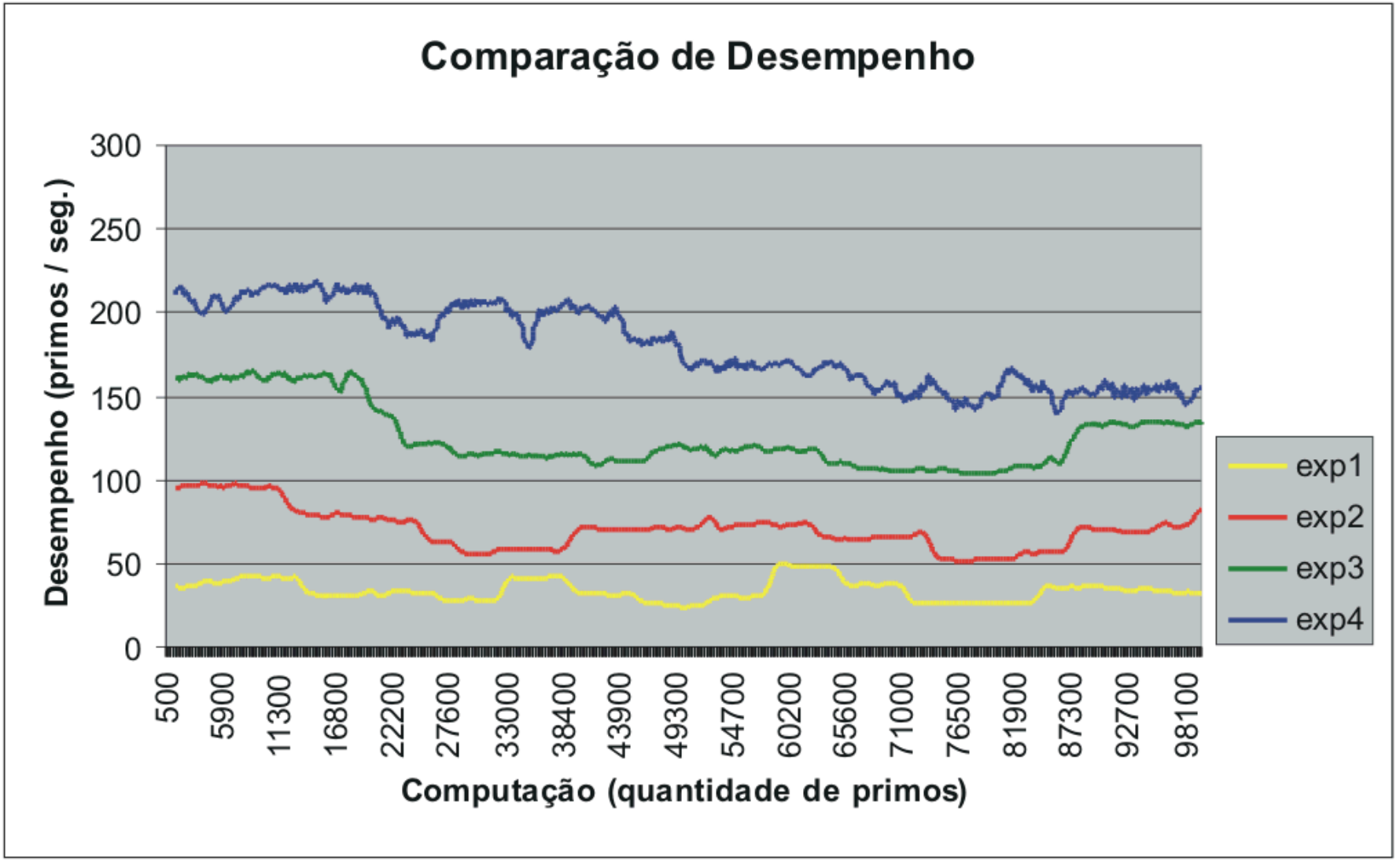
Número Fixo de Computadores

Laboratório de Redes de Computadores



Número Fixo de Computadores

Laboratório de Redes de Computadores



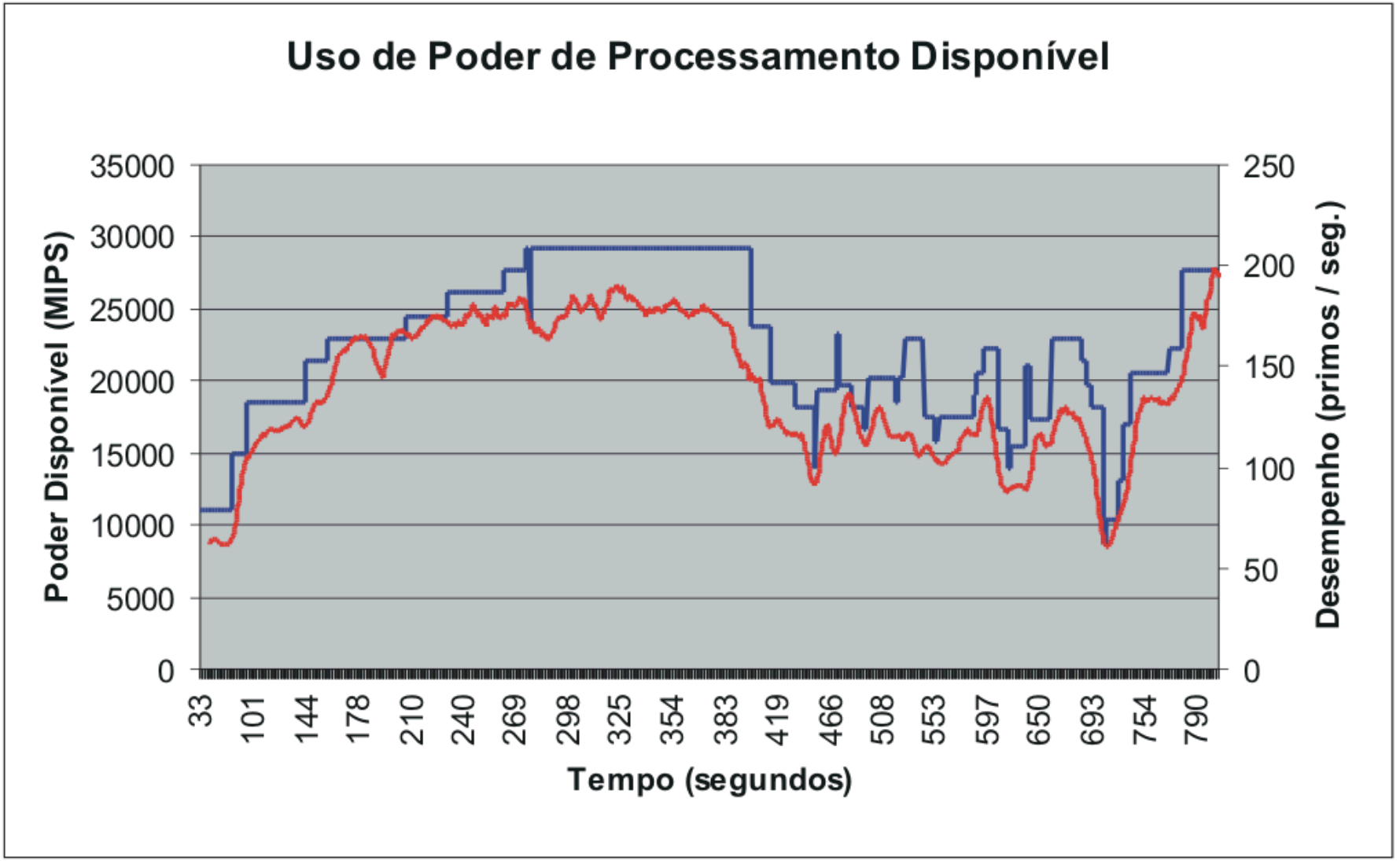
Número Variável de Computadores

Laboratório de Redes de Computadores

- Cálculo de 100.000 números primos
- Foi realizado um experimento
  - 10 máquinas heterogêneas iniciadas e paradas durante a execução da aplicação
  - Poder de processamento: 5.538 a 29.298 MIPS
- Mecanismos utilizados
  - Reescalonamento e redistribuição de atividades aos recursos disponíveis
  - Recuperação de atividades
- Resultado obtido
  - Grande correlação entre o poder computacional disponível e o desempenho da aplicação
  - Aproveitamento da computação realizada
- Avaliação
  - O middleware é dinamicamente adaptável à variabilidade da grade e prontamente utiliza os recursos disponíveis

Número Variável de Computadores

Laboratório de Redes de Computadores



Laboratório de Redes
de Computadores

2. Algoritmos de escalonamento para execução de tarefas dependentes

O problema de escalonamento

Laboratório de Redes de Computadores

- **Escalonamento: distribuir tarefas entre recursos para obter a execução mais eficiente possível em termos de uma função objetivo.**
  - Entradas básicas em um sistema heterogêneo:
    - T={T1, T2, ..., Tn}, onde Ti={c, D}: Ti é uma tarefa a ser executada, com seu custo de computação (c) e conjunto dependências de dados entre tarefas (D).
    - R={R1, R2, ..., Rm}. onde Ri={p, L}: Ri é um recurso disponível, com sua capacidade de processamento e conjunto de capacidades dos links de comunicação com outros recursos (L).
  - Saída: mapeamento de T em R.
- **Escalonador: entidade responsável pela realização do escalonamento.**
  - Aguarda a submissão de tarefas e processos por parte dos usuários, para então realizar o escalonamento.

Escalonamento de tarefas dependentes

Laboratório de Redes de Computadores

- **Tarefas possuem dependências de dados que devem ser consideradas durante o escalonamento**
  - Dependências de dados significam tempo de comunicação (custo de comunicação)
- **Problema NP-Completo: não há algoritmo conhecido que gere solução ótima em tempo polinomial**
  - Heurísticas: execução rápida, nenhuma garantia (não experimental) quanto à qualidade da solução obitda.
  - Meta-heurísticas: algoritmos genéticos (GA), procedimento guloso adaptativo de busca (greedy randomized adaptive search procedure – GRASP):
    - execução não tão rápida quanto heurísticas: depende do número de iterações imposto pelo programador.
    - nenhuma garantia quanto à qualidade da solução obtida: ótimos locais são freqüentemente a solução final.
  - Programação linear: tempo de execução e qualidade da solução dependem do relaxamento de restrições e da redução do número de variáveis.
  - Algoritmos de aproximação: algoritmos difíceis de obter para problemas genéricos, porém oferecem garantia da qualidade da

Heurísticas – algumas técnicas

Laboratório de Redes de Computadores

- **List Scheduling**
  - Criar lista de tarefas de acordo com critério de prioridade.
  - Selecionar tarefa da lista de acordo com estratégia de seleção de tarefas.
  - Seleciona recurso para tarefa de acordo com estratégia de seleção de recursos.
- **Clustering**
  - Técnica de duas fases
    - 1. Agrupamento: grupos de tarefas são criados de acordo com estratégia, considerando-as como escalonadas em um cluster homogêneo virtual com um número ilimitado de recursos.
    - 2. Mapeamento: cada cluster de tarefas é escalonado em um recurso, de acordo com estratégia de seleção de recursos.
- **Task Duplication**
  - Consiste em escalonar a mesma tarefa em mais de um recurso, buscando redundância na execução.
  - Em caso de falha em um recurso, outro pode retornar o resultado.
  - Se não há falha, o primeiro resultado retornado pode ser utilizado e a outra execução pode ser abortada.

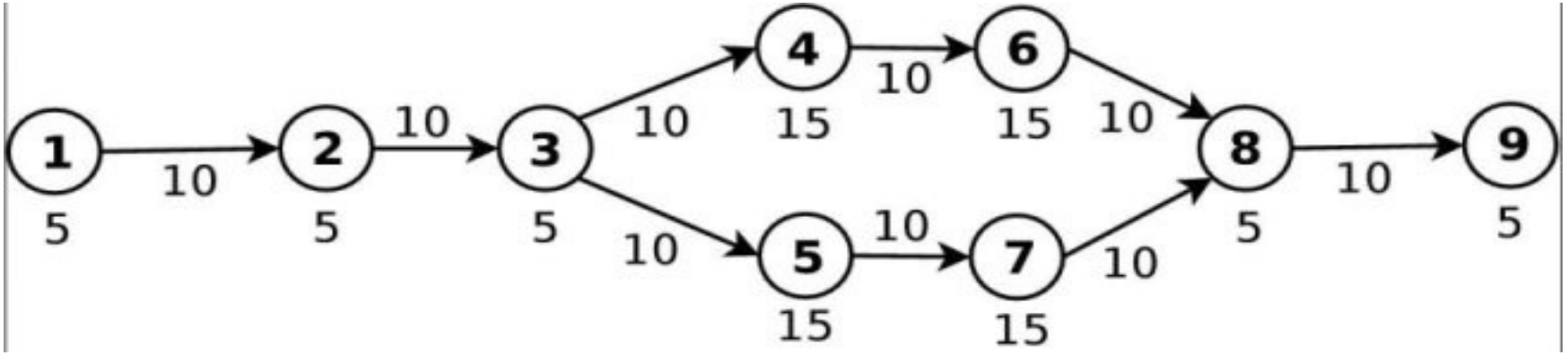
Escalonamento de tarefas dependentes em grades computacionais

Laboratório de Redes de Computadores

- **Workflows**
  - Tarefas dependem de dados computados por seus predecessores.
  - É necessário considerar o tempo de comunicação entre tarefas.
- **Dificuldades em grades**
  - Ambiente dinâmico.
  - Recursos compartilhados.
  - Heterogeneidade.
- **Escalonamento de tarefas pode ter diferentes funções objetivo**
  - Minimizar o tempo de execução, maximizar o *throughput, m*inimizar o tráfego na rede, etc.
  - Nós focamos no primeiro.
- **Escalonamento estático e dinâmico**
  - Estático: escalona todas as tarefas do workflow antes de iniciar a execução do processo.
  - Dinâmico: escalona parte das tarefas do workflow, aguarda execução e utiliza dados da execução para tomar decisões no escalonamento do restante do processo.

Representação das dependências

- **Um processo é representado por um grafo acíclico direcionado (directed acyclic graph - DAG)**
  - Custos de computação e custos de comunicação



- **Tarefa 8 só pode iniciar sua execução após as tarefas 6 e 7 terminarem e enviarem os dados através da rede**
- **O custo de comunicação de tarefas no mesmo recurso é zero**
  - Agrupar tarefas dependentes para reduzir o tempo de comunicação: técnica de *clustering (agrupamento de tarefas)*

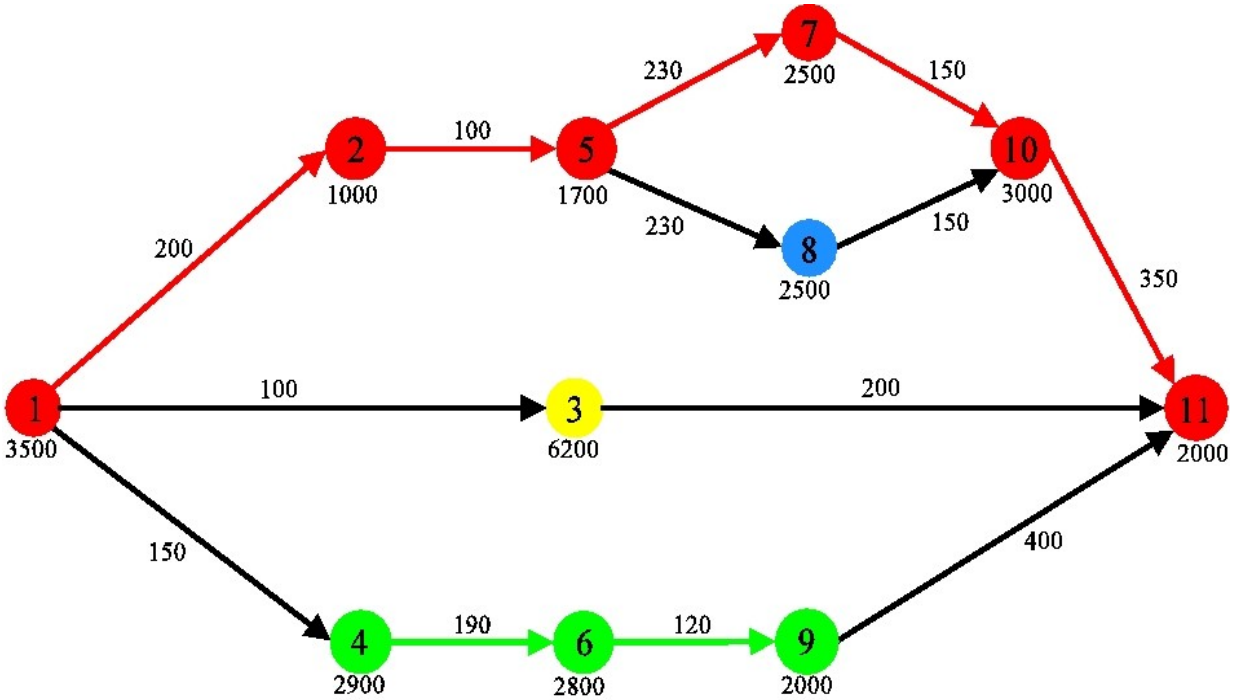
Clustering: visão geral de um algoritmo estático

Laboratório de Redes de Computadores

- **1. WHILE (Existem tarefas não escalonadas)**
- **2. *cluster* <- seleciona_próximo_cluster()**
- **3. *recurso* <- seleciona_melhor_recurso(*cluster*)**
- **4. Escalona *cluster* em *recurso***
- **5. END WHILE**
- **6. Envia tarefas para execução**

Clustering: O algoritmo Path Clustering Heuristic (PCH) [1]

- **Tarefas em um caminho comum são agrupadas**
  - Tarefas em um mesmo cluster são escalonadas no mesmo recurso

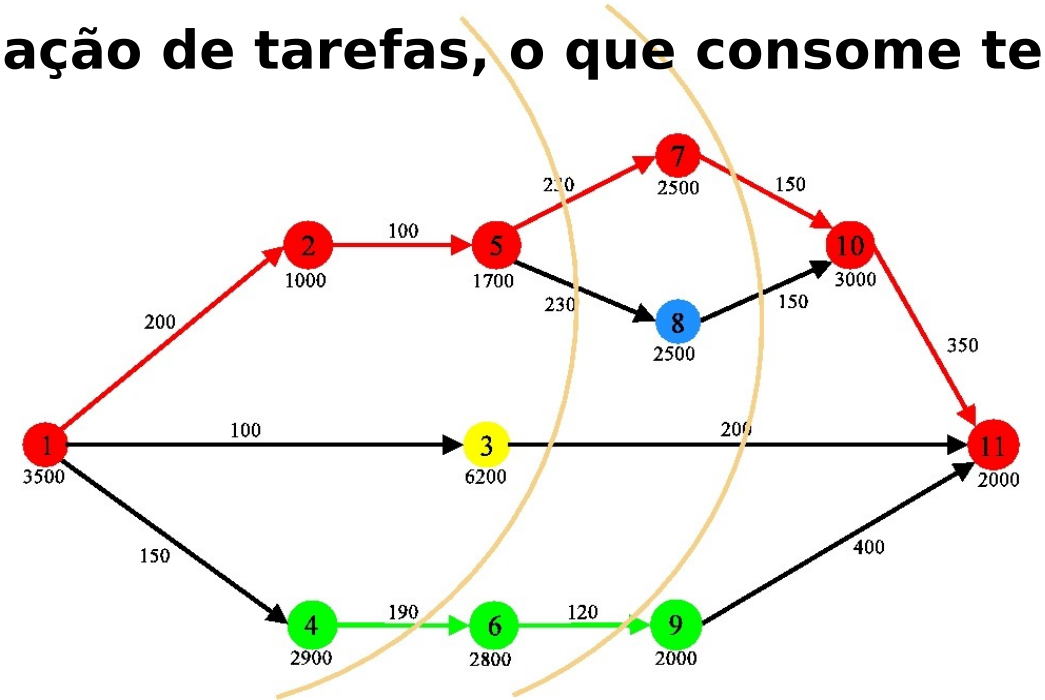


- **Clusters vermelho e verde: comunicação é zero com as tarefas escalonadas no mesmo recurso**
  - Problema: recursos são compartilhados e podem não ter toda sua capacidade de processamento disponível durante toda a execução do cluster
  - Solução: escalonamento dinâmico.

Escalonamento dinâmico de tarefas [1]

Laboratório de Redes de Computadores

- **E se o dono do recurso inicia a execução de processos independentes das tarefas controladas pela grade?**
  - Se todas as tarefas foram enviadas para execução, uma queda de desempenho pode atrasar a execução do processo.
  - Uma abordagem dinâmica pode ser usada para enviar tarefas à medida que seus predecessores terminam de executar.
- **A cada grupo de tarefas executado (*round* de execução), o escalonador tem uma nova visão dos recursos com conhecimento do desempenho atual.**
- **Evita realocação de tarefas, o que consome tempo e largura de banda.**



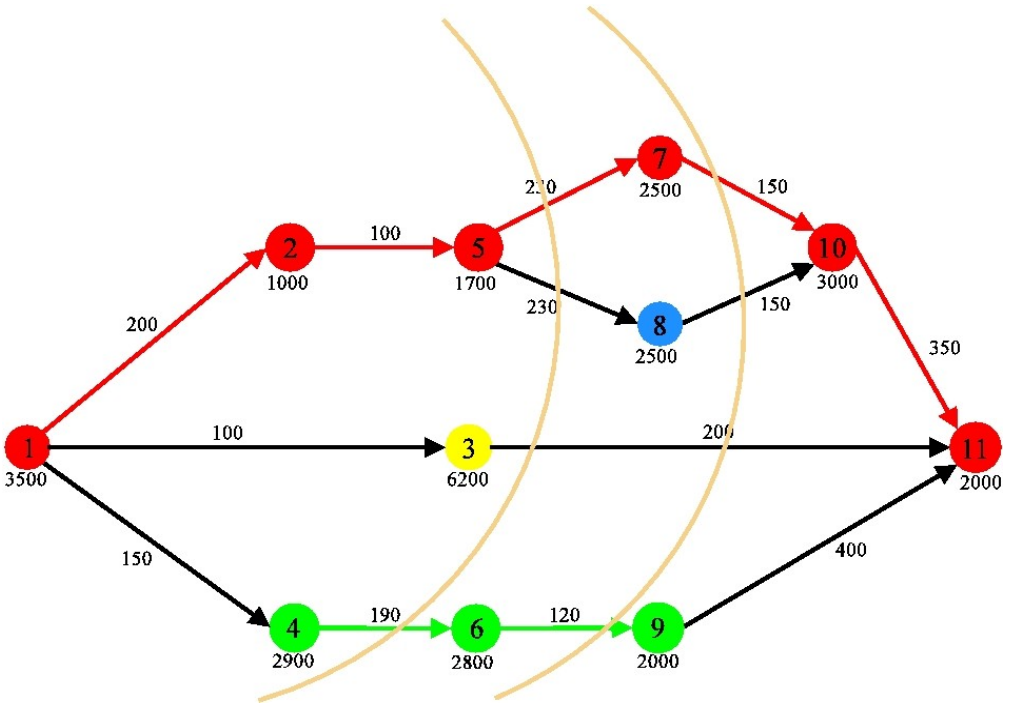
Visão geral de um algoritmo de escalonamento dinâmico

Laboratório de Redes de Computadores

1. Escalona DAG *G* usando o algoritmo PCH estático
2. WHILE not(todas as tarefas de *G* terminaram)
3. Seleciona tarefas para executar de acordo com uma política.
4. Envia tarefas deste *round* para execução.
5. Avalia desempenho dos recursos.
6. Re-escalona tarefas se necessário.
7. ENDWHILE

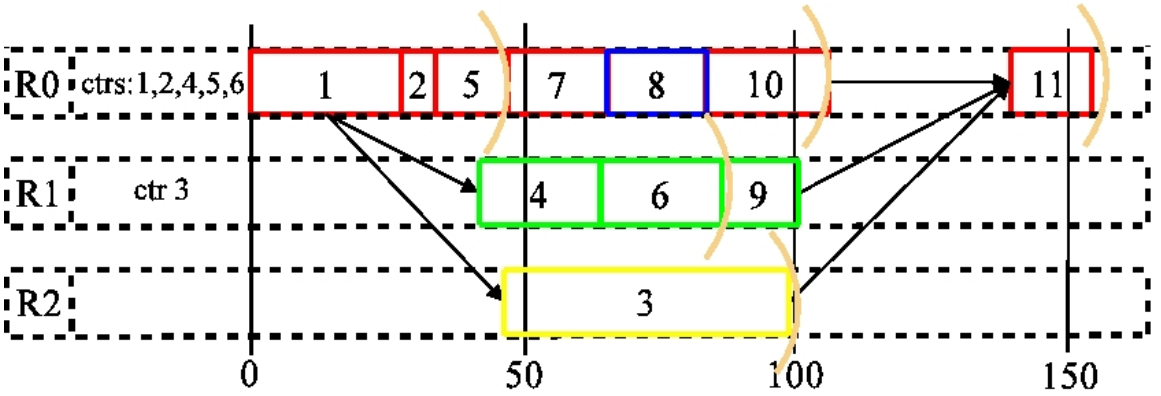
Escalonamento dinâmico de tarefas [1]

- **Escalonador recebe *feedback* dos recursos**
  - Comparando o tempo estimado de execução com o tempo real de execução mostra a capacidade atual do recurso.
- **Utilizando o *feedback* o escalonador pode decidir quais tarefas continuarão com o escalonamento inicial e quais devem ser re-escalonadas.**
- **Estratégia baseada em *rounds* de execução**
  - Em cada *round* o escalonador envia algumas tarefas para execução, de acordo com um critério, e aguarda o *feedback* para tomar decisões.



Escalonamento dinâmico adaptativo [2]

- ***Rounds* de execução são adaptados de acordo com o desempenho dos recursos**
  - Se o desempenho não é bom, o algoritmo reduz o tamanho do round, fazendo-o ter menos tarefas.
  - Utiliza informação passada (outros workflows) para adaptar o tamanho do *round* para novos processos.
  - Adapta o tamanho do *round* de acordo com o tamanho das tarefas sendo escalonadas: tarefas maiores implicam em menor número de tarefas em cada *round*.
  - Tamanho do *round* é independente em cada recurso.

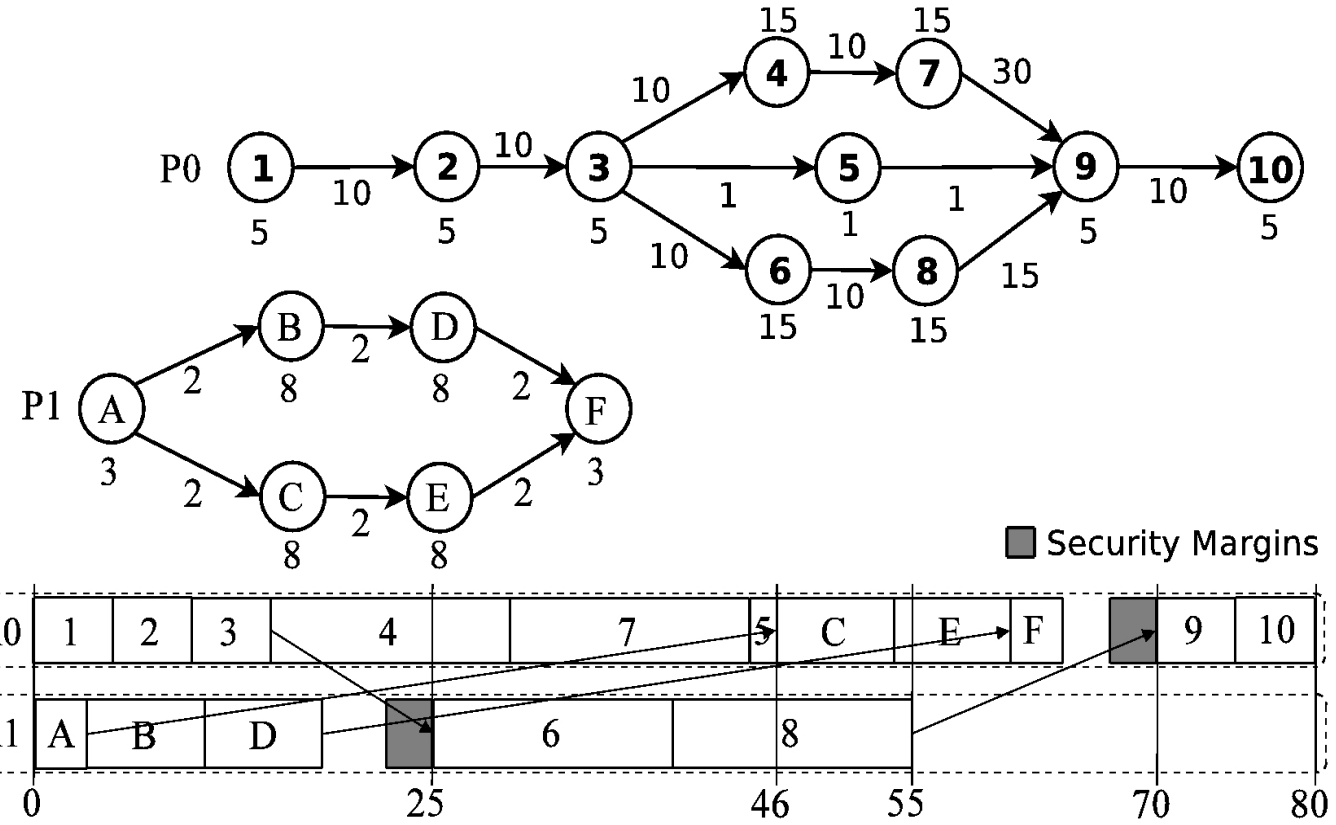


- **Resource 0: 3 rounds**
- **Resource 1: 1 round**
- **Resource 2: 1 round**

Procura e preenchimento de espaços [3]

Laboratório de Redes de Computadores

- **Dependências geram espaços entre a execução de tarefas, como por exemplo antes da tarefa 11 no slide anterior.**
  - Outros processos podem utilizar esse espaço livre para executar tarefas, minimizando o tempo de execução geral dos processos.
  - Algoritmos para preenchimento de espaços, combinando adaptação e considerando variações de performance (utilizando margens de segurança, por exemplo), podem aproveitar esses espaços em grades.



Métricas de desempenho

Laboratório de Redes de Computadores

- ***Makespan*: é o "tamanho" do escalonamento obtido, ou seja, o tempo de término da última tarefa do workflow.**
- **Schedule Length Ratio (SLR):**

$$SLR = \frac{makespan}{\sum_{n_i \in CP} \frac{instructions_{n_i}}{power_{best}}}$$

- onde: CP é o conjunto de tarefas (nós) pertencentes ao caminho crítico do DAG e *power_best* é a capacidade de processamento do melhor recurso disponível.
- Quantas vezes o makespan resultante é maior que o caminho crítico do processo quando executado no melhor recurso.

Métricas de desempenho

Laboratório de Redes de Computadores

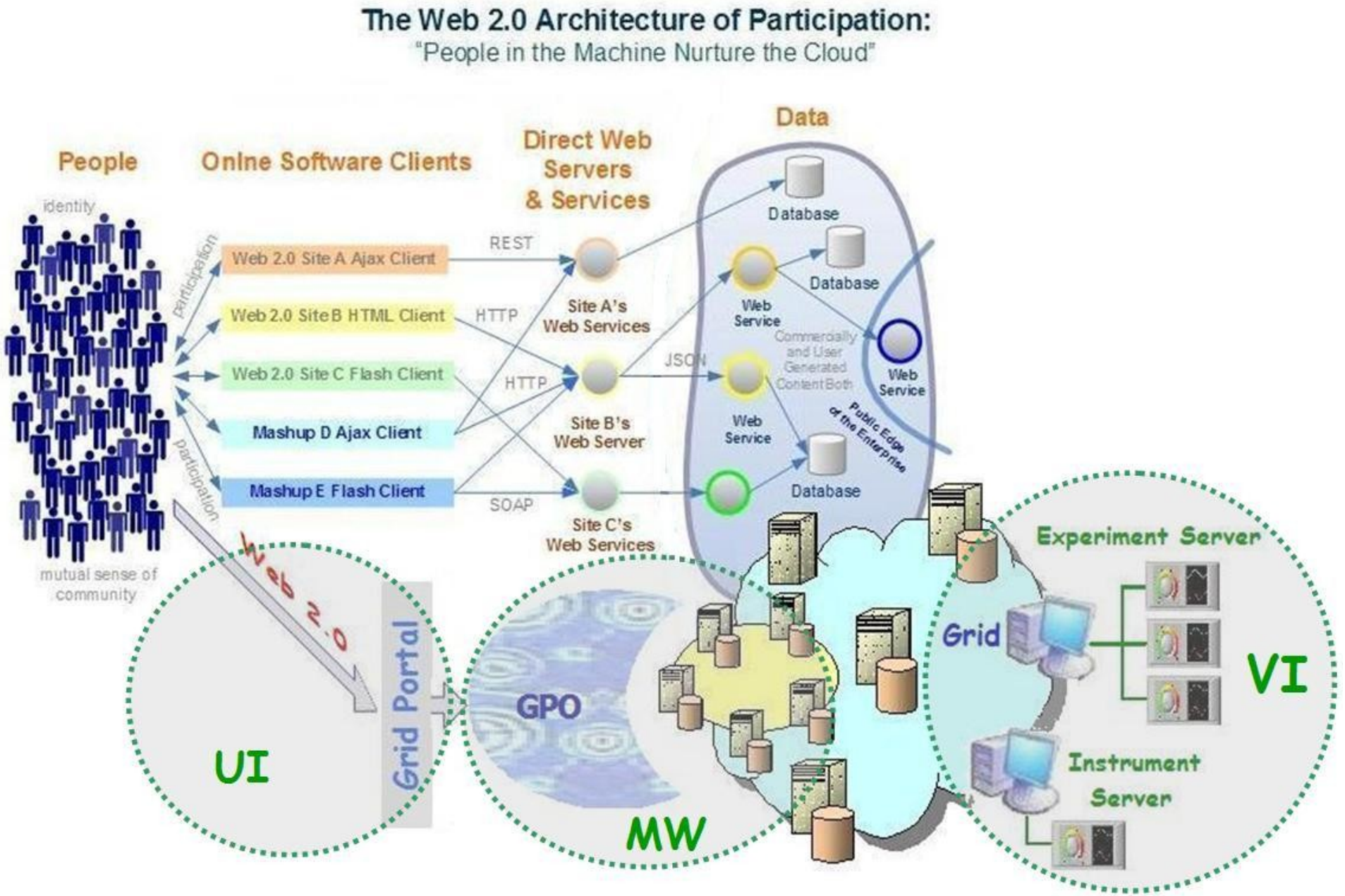
- **Speedup:**

$$Speedup = \frac{\sum_{n_i \in V} \frac{instructions_{n_i}}{power_{best}}}{makespan}$$

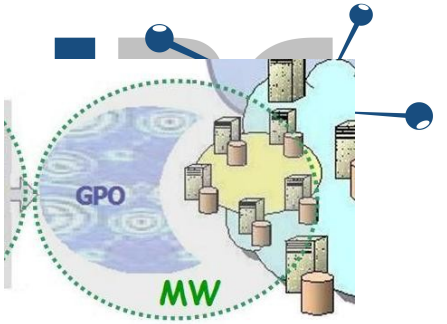
- onde: V é o conjunto de tarefas (nós) do DAG e *power_best* é a capacidade de processamento do melhor recurso disponível.
- Quantas vezes mais rápida é a execução com o escalonamento dado quando comparada à execução seqüencial de todas as tarefas no melhor recurso disponível.

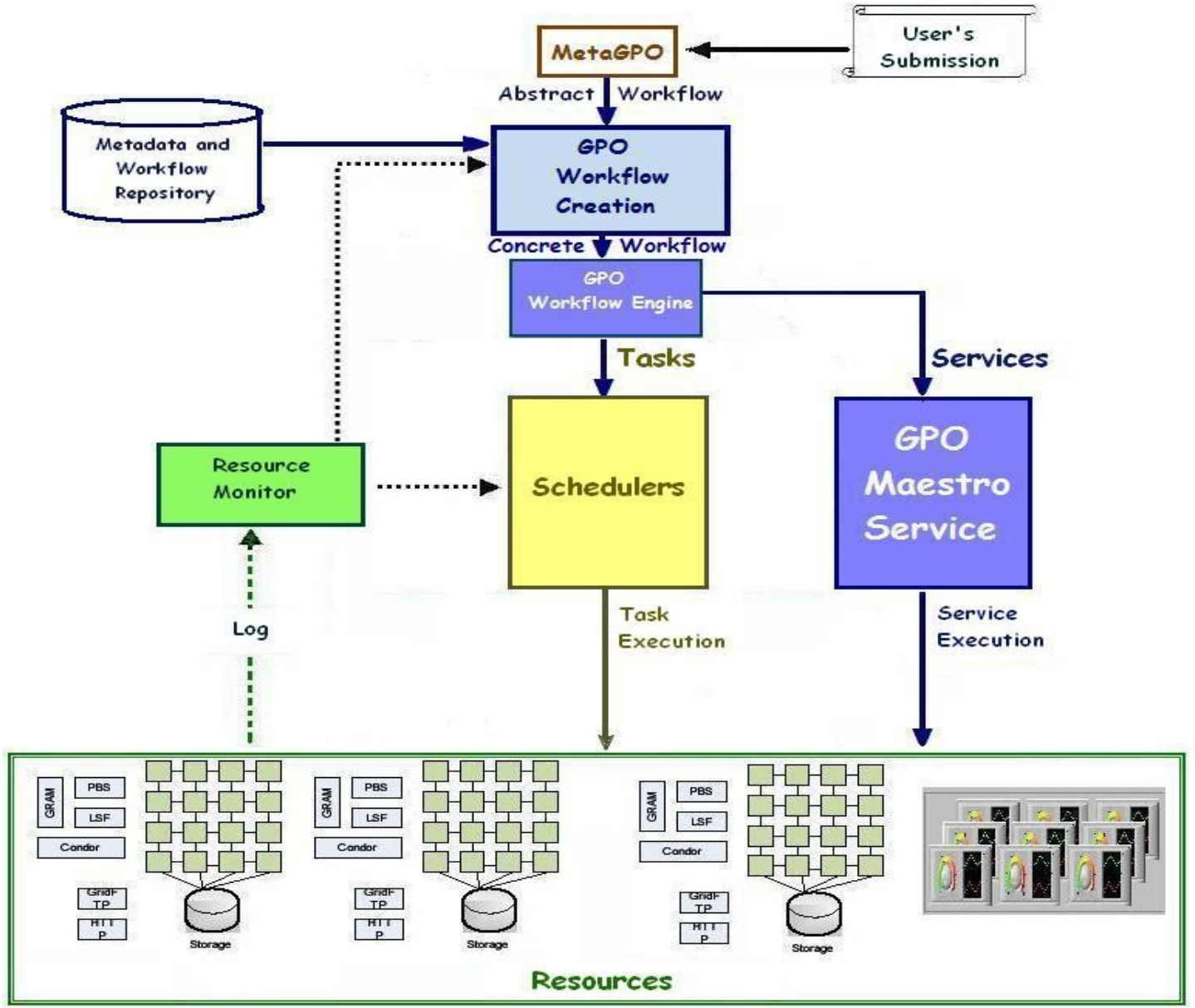
3. Ambientes de Grade no contexto da Web 2.0 Colaborativa para E-Ciência

GEOP Architecture



GEOP: Metaworkflow Management (MW)





GEOP: Metaworkflow Management (MW)

Laboratório de Redes de Computadores

- **MetaGPO**

  Generates an abstract workflow with resource classes indication, which permits to the GPOWC choose the better resources for concrete workflow generation.

- **GPO Workflow Creation (GPOWC)**

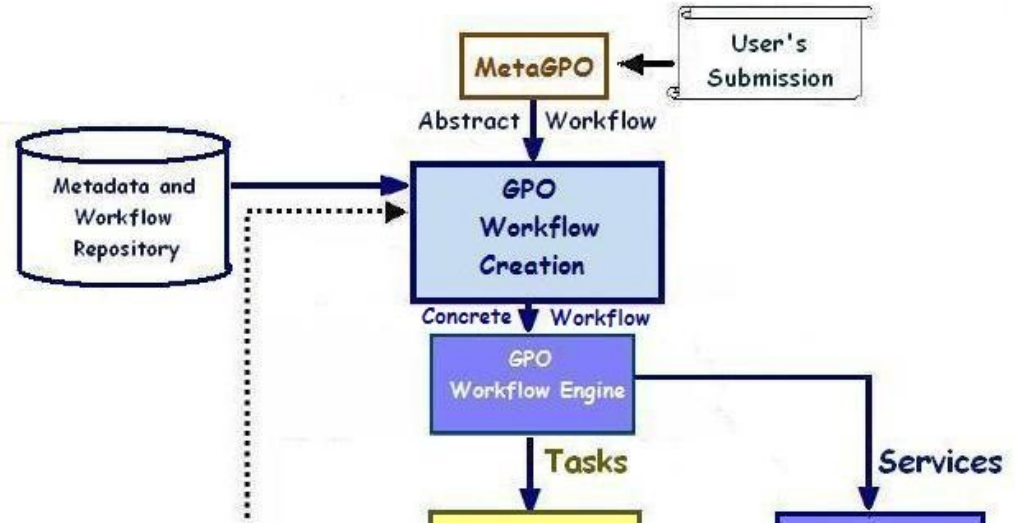
  Verifies the resources available in Grid, consults the repository for make the expansion of the abstract workflow, and generates the concrete workflow using the GPO Langua

- **GPO Workflow Engine**

  Puts the concrete workflow in

  actions, and manages

  its execution.



Conclusão

- Grade Peer-to-Peer (descentralizado)
- Tolerância a Faltas
- Monitoração (CPU, Memória, Comunicação,...)
- Estimativa de Processamento, Memória, Comunicação, ...
- Vários DAGs
- Multicore
- E-Ciência (Colaboração)
- Workflows (Governo Eletrônico, ...)
- Grades de Serviços Web
- .....

Referências

[1] F. CICERRE; E. R. M. MADEIRA e L. E. BUZATO. *Structured process execution middleware for grid computing*. Concurrency and Computation: Practice and Experience. John Willey & Sons, v. 18, n. 6, p. 581-594, 2006.

[2] L. F. Bittencourt e E. R. M. Madeira. *A Dynamic Approach for Scheduling Dependent Tasks on the Xavantes Grid Middleware.* 4th ACM International Workshop on Middleware for Grid Computing (MGC), Melbourne, Austrália, nov., 2006.

[3] L. F. Bittencourt e E. R. M. Madeira. *A performance-oriented adaptive scheduler for dependent tasks on grids.* Concurrency and Computation: Practice and Experience. John Willey & Sons, Online, 2007.

[4] L. F. Bittencourt e E. R. M. Madeira. *Fulfilling Task Dependence Gaps for Workflow Scheduling on Grids. 3rd* IEEE International Conference on Signal-Image Technology & Internet Based Systems (SITIS), Shanghai, China, dez., 2007.

[5] C. SENNA e E.R.M. MADEIRA. *A middleware for instrument and service orchestration in computational grids*. 7th IEEE International