Modelos de concorrência

Programação concorrente

Tiago Salmito

tsalmito@inf.puc-rio.br

Conteúdo

- Modelos básicos de concorrência
  - Classificação
  - Características
  - Comparação
- Modelos híbridos de concorrência
  - Definição
  - Classificação
  - Características
  - Exemplos

Concorrência e paralelismo

- Concorrência
  - Propriedade de programas em que mais de uma linha de execução (ou computação) ocorre simultaneamente
    - Fazem progresso, mas não necessariamente ao mesmo tempo
    - Potencialmente interagindo entre si
    - Competem por recursos
- Paralelismo
  - Propriedade de sistemas em que pelo menos duas computações podem ser realizadas ao mesmo tempo
- Problema
  - Processo vs processador
    - Caso típico: Tarefas concorrentes >> processadores

Gerenciamento de tarefas concorrentes

- Gerenciamento de tarefas concorrentes
  - Preemptivo
    - Tarefas concorrentes são executadas de forma intercalada
    - Tarefas podem ser interrompidas antes do término
    - Escalonador decide qual o tempo e a ordem de execução de tarefas
  - Cooperativo
    - Tarefas concorrentes passam o fluxo de controle entre si de forma explícita
    - Tarefas não podem ser interrompidas (atômicas)
- Gerenciamento de contexto de tarefas (ou gerenciamento de pilha)
  - Automático
    - Contexto local de uma tarefa concorrente é mantido durante toda a computação
      - Sobrecarga de troca de contexto
  - Manual
    - Contexto local é perdido entre tarefas concorrentes
      - Deve ser explicitamente gerenciado pelo programador

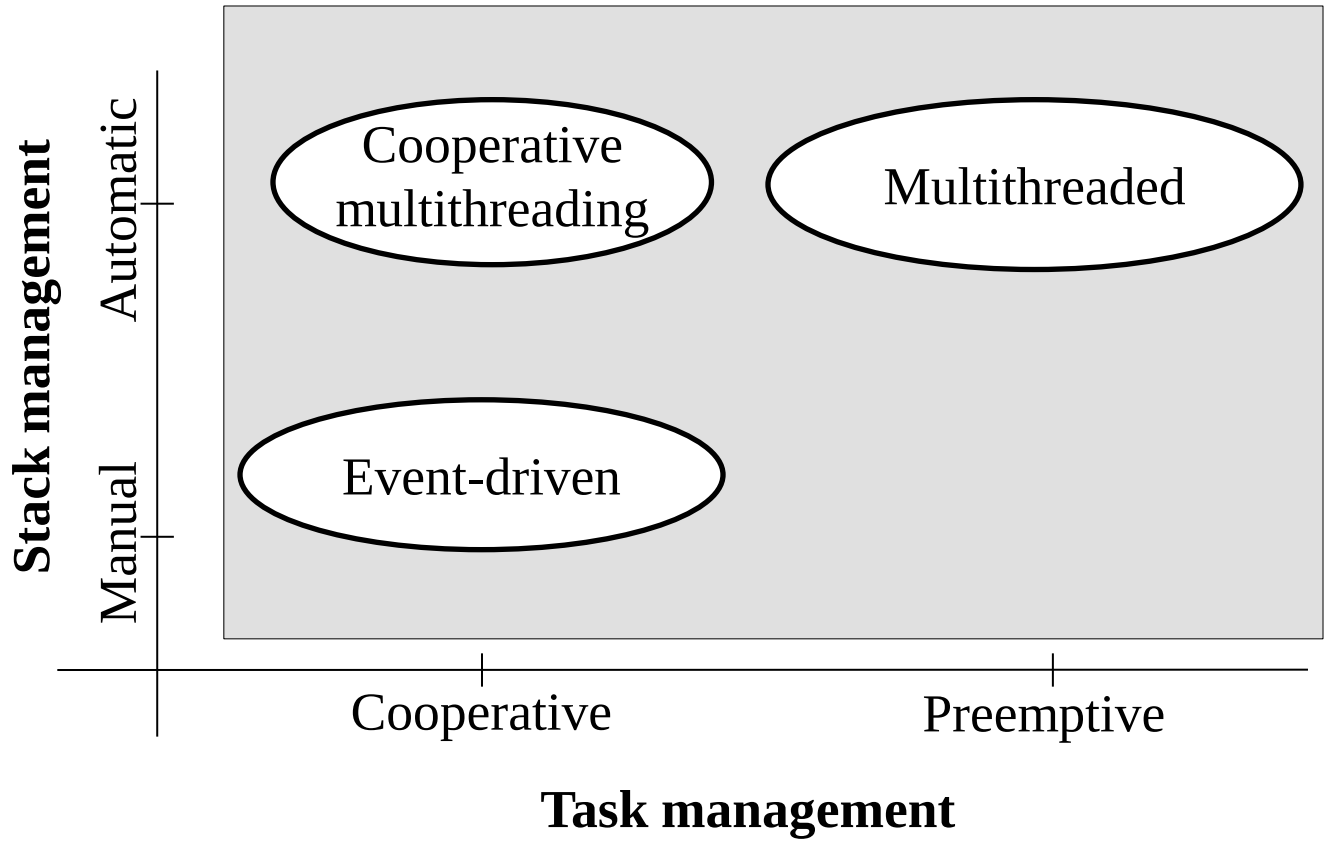
Classificação de modelos básicos de concorrência

- Baseados em threads / processos
  - Natural para o programador
    - Fluxos de controle concorrentes
  - Gerenciamento de pilha automático
    - Pilha de procedimentos é guardada e restaurada automaticamente durante a troca de contexto pelo escalonador
  - Gerenciamento de tarefas preemptivo
- Orientados a eventos
  - Natural para o processador
    - Modelo de interrupções de hardware
  - Máquinas de estado
    - Contexto global
  - Gerenciamento de tarefas cooperativo
  - Gerenciamento de pilha manual
    - Troca de mensagens assíncronas
      - Eventos

Classificação de modelos básicos de concorrência

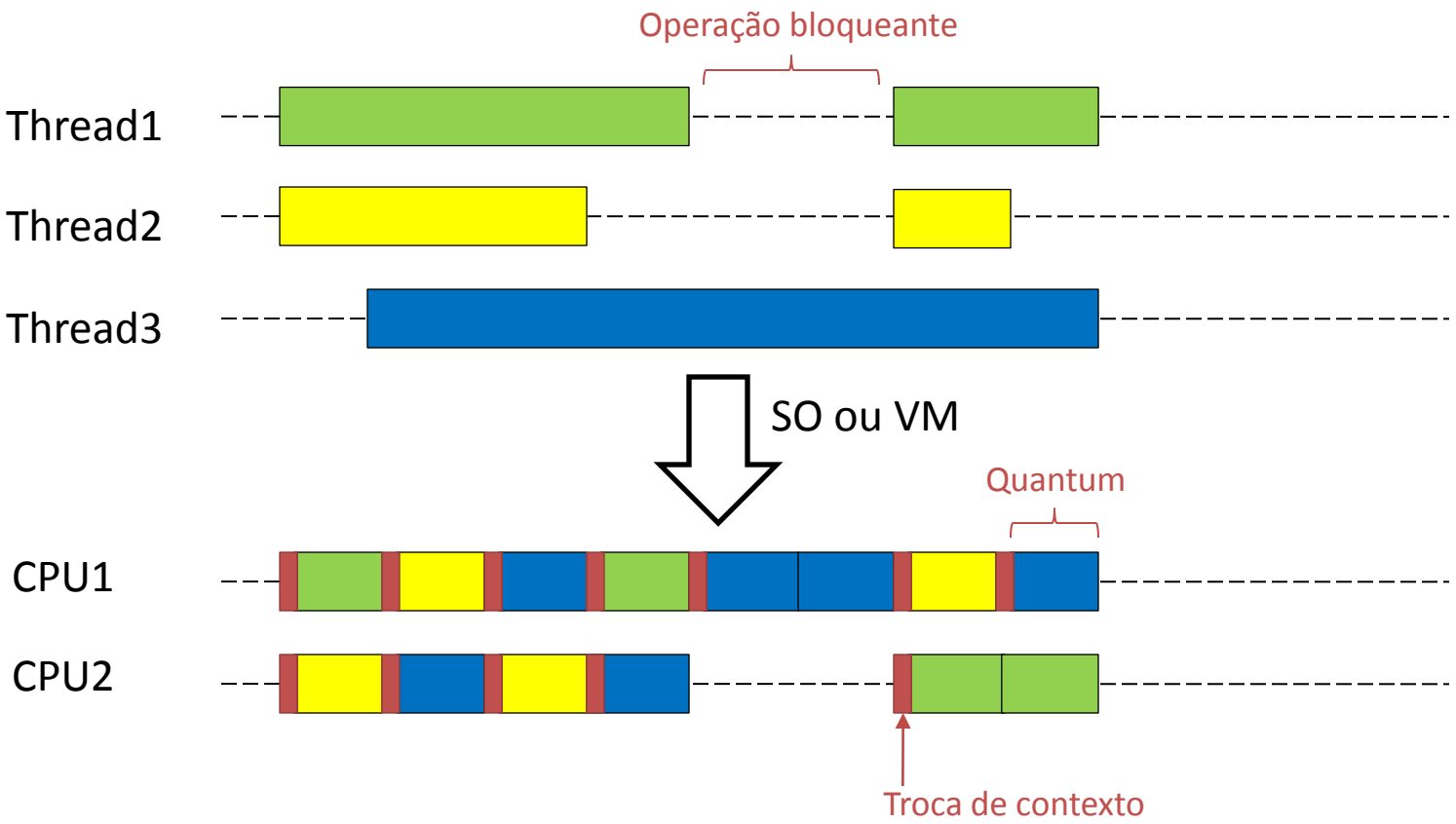
- Caso especial: Multitarefa cooperativa
  - Também chamados de threads leves
    - Threads em nível de usuário, Corotinas, Fibras
  - Gerenciamento de pilha automático
  - Gerenciamento de tarefas cooperativo
  - Multitarefa cooperativa simétrica
    - Provê uma única operação para troca explícita de controle entre tarefas concorrentes (*Transfer*, *SwitchTo*)
  - Multitarefa cooperativa assimétrica
    - Operação para passar o controle para uma tarefa (*resume*)
    - Operação para retornar o controle para a tarefa que a chamou (*yield*)

Classificação de modelos básicos de concorrência

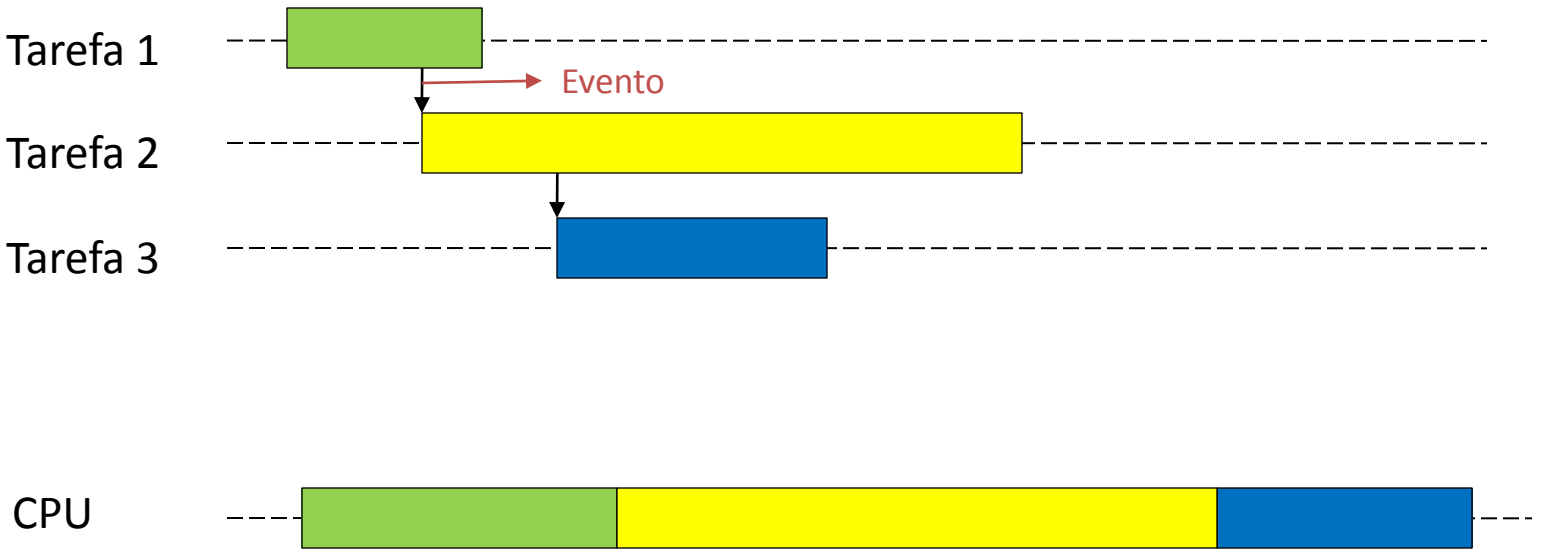


* A. Adya, J. Howell, M. Theimer, W. J. Bolosky and J. R. Douceur. *Cooperative Task Management Without Manual Stack Management.* (2002)

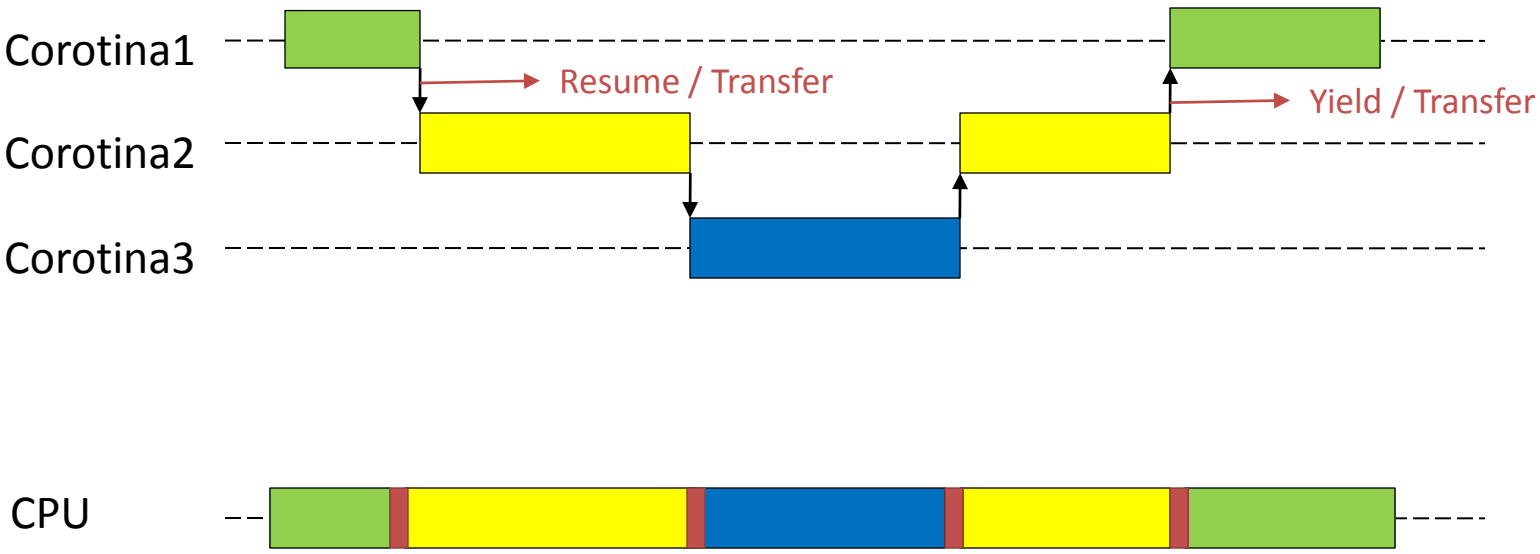
Multitarefa Preemptiva



Eventos

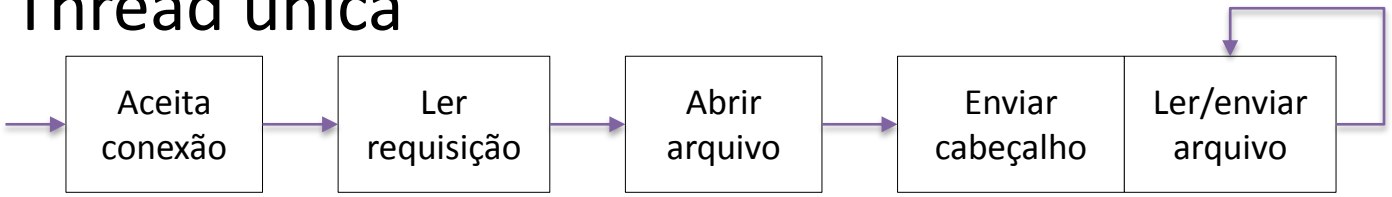


Multitarefa cooperativa

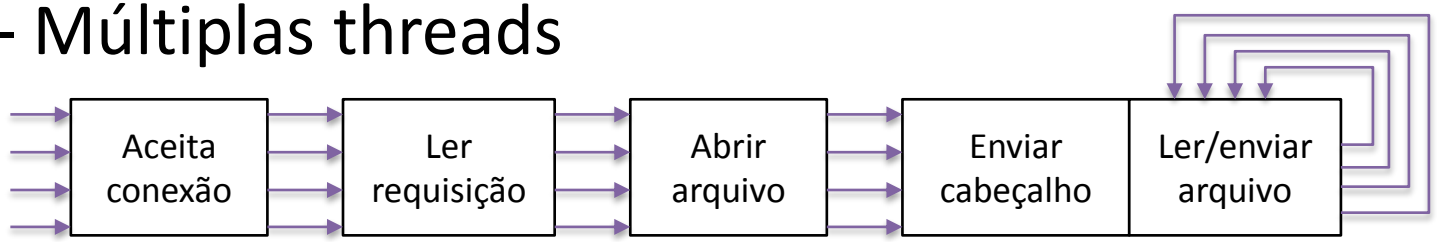


Estudo de caso: Servidores web

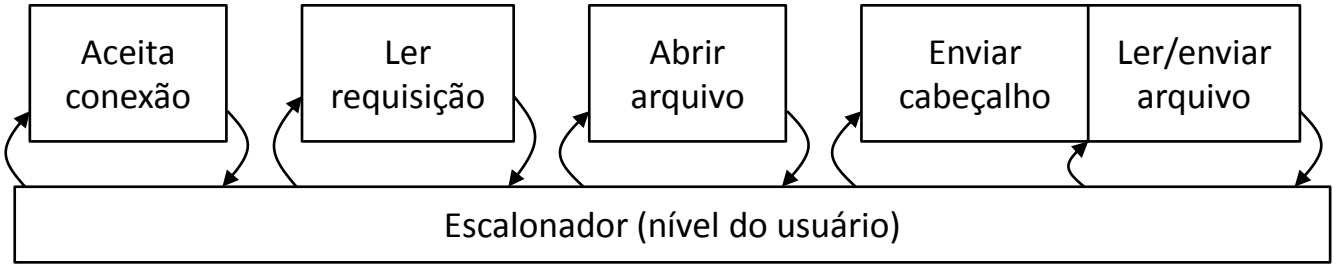
– Thread única



– Múltiplas threads



– Orientado a eventos



* Pai, V. et al.: *Flash: An efficient and portable web server.* (1999)

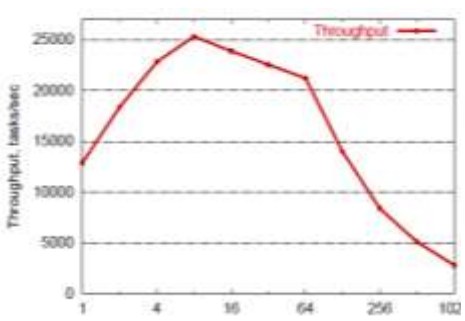
Dualidade thread-evento

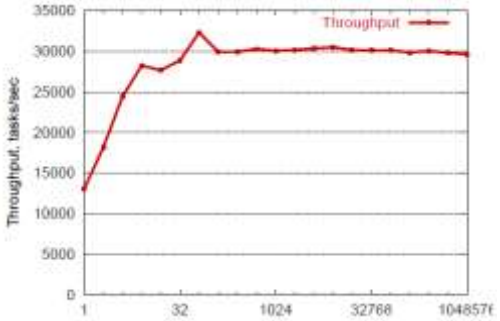
Orientados Procedimento (threads)		Orientados a mensagens (eventos)
Chamada de procedimento	↔	Mandar mensagem e aguardar resposta
Escalonador	↔	Loop de mensagens
Retorno de procedimento	↔	Mandar resposta
Continuação de procedimento	↔	Tratador de mensagens
Variáveis de condição e travas	↔	Aguardando mensagens

* Lauer, H.C., Needham, R.M.: *On the duality of operating system structures.* (1979)

Críticas sobre threads

- Escalonador escondido
  - Política fixa e geral
    - Prioridades
  - Deadlocks, livelocks e starvation
  - Condições de corrida
- Sobrecarga de concorrência
  - Criação
  - Troca de contexto
  - Sincronização
- Esgotamento de Vazão
  - Degradação grosseira

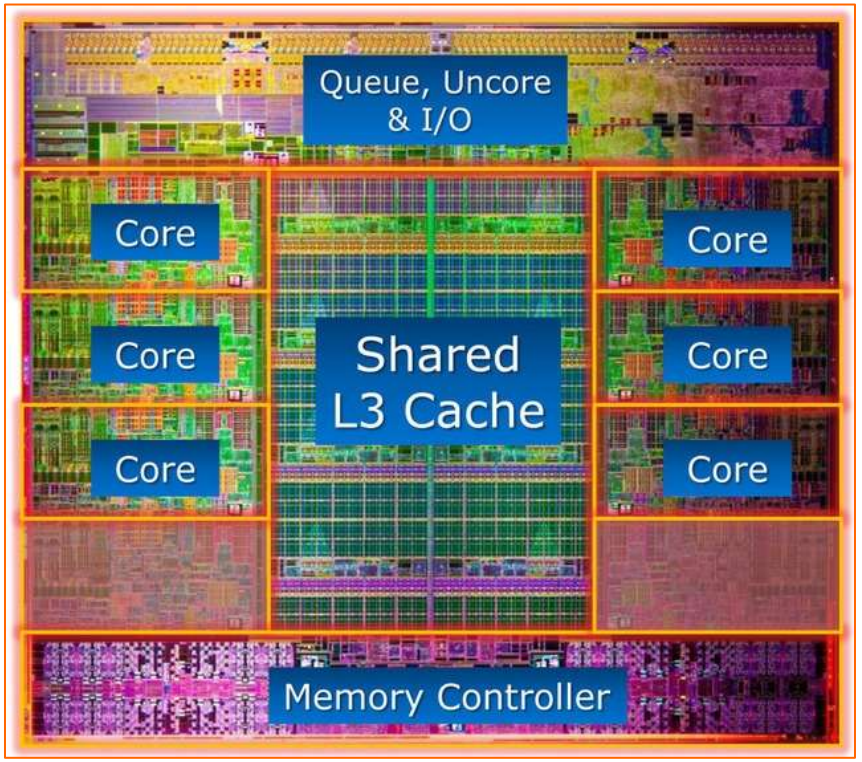




* Ousterhout, J.: *Why threads are a bad idea (for most purposes).* (1996)

Críticas sobre eventos

- Paralelismo real
- Operações não-bloqueantes
  - Tratadores de eventos devem ser atômicos e não podem bloquear
- Inversão de controle
  - Fluxo de controle é definido pela ordem se execução dos eventos e não pelo escalonador do sistema
  - Complexidade de manutenção e depuração
- Fluxo de controle é expresso de forma mais natural com threads
  - Chamada e retorno de procedimentos



* Behren, R., Condit, J., and Brewer, E.: . *Why events are a bad idea (for high-concurrency servers).* (2003)

Modelos básicos de concorrência (resumo)

Baseados em threads

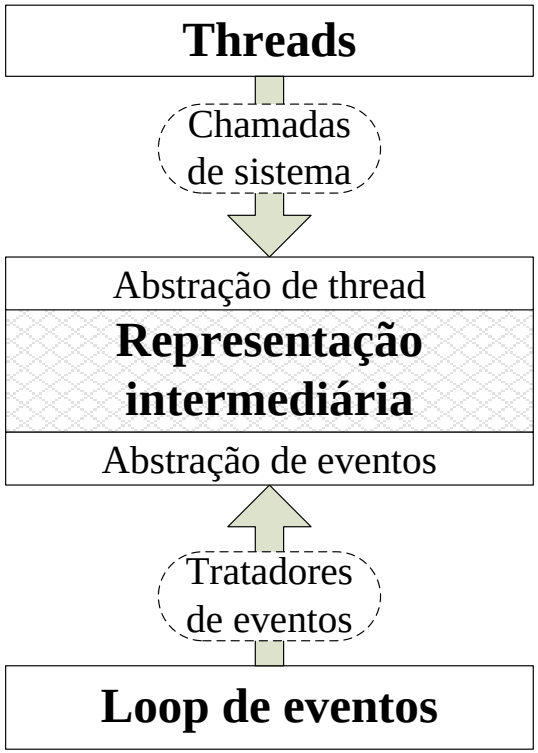
- Preempção
- Sincronização
- Operações bloqueantes
- Natural para o programador
  - Porém introduz problemas de sincronização
- Prioriza latência

Orientados a eventos

- Cooperação
- Inversão de controle
  - Perda da pilha local
- Natural para o computador
  - Modelo de interrupções
- Prioriza vazão

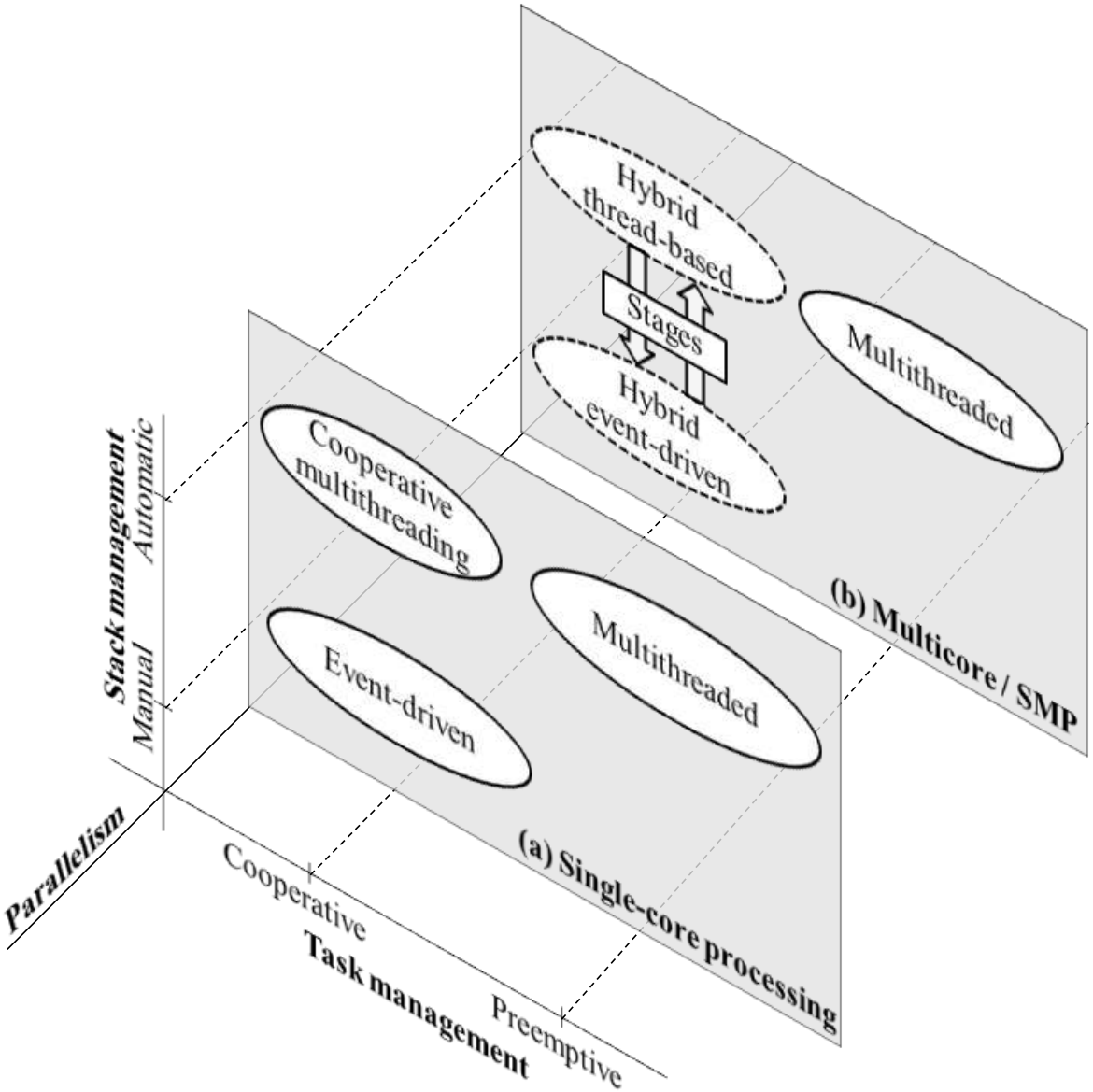
Concorrência Híbrida

- Modelos de concorrência que combinam thread e eventos
  - Paralelismo real
- Ambiente de programação:
  - Orientado a eventos
    - Há inversão de controle
    - Permitem múltiplos loops concorrentes
  - Baseado em threads
    - Threads no nível de usuário
    - Compartilham cooperativamente um pool de threads
  - Baseado em estágios
    - Mescla entre os dois modelos
    - Pipelines



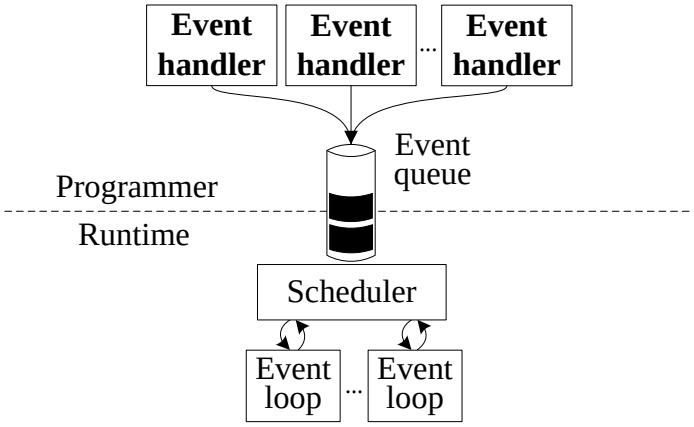
* Li, P., Zdancewic, S.: *Combining events and threads for scalable network services implementation and evaluation of monadic, application-level concurrency primitives*. (2007)

Modelos híbridos de concorrência



Concorrência híbrida orientada a eventos

- Ambiente orientado a eventos com múltiplos loops concorrentes
  - Sem operações bloqueantes
- Fila de eventos compartilhada
- Escalonador
  - Sincronização de eventos
  - Controla o consumo de eventos
- Simétricos ou assimétricos

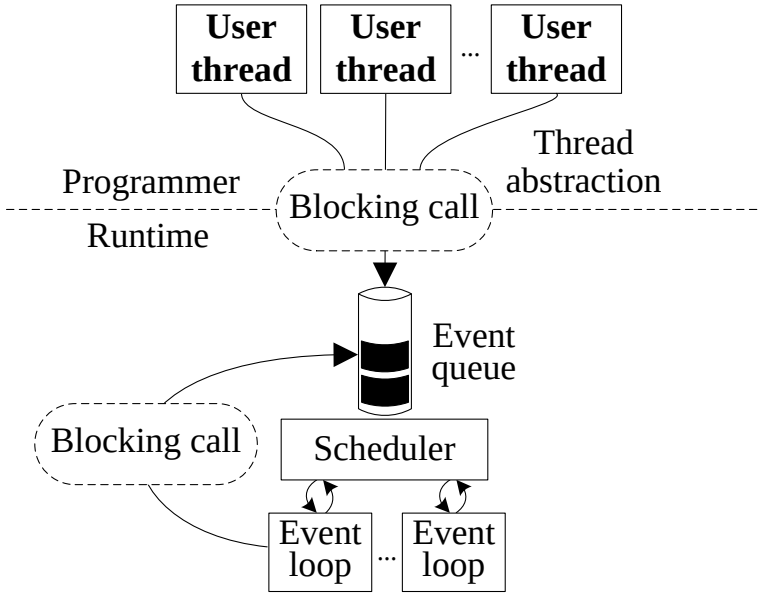


Eventos híbridos: Algumas abordagens

- Libasync-smp
  - Sincronização por cores de eventos
    - Eventos de mesma cor não podem executar concorrentemente
    - Cores são atribuídas manualmente pelo programador
  - Pool de thread estático (tipicamente uma por CPU)
- InContext
  - Sincronização por níveis de permissão (ou contextos)
    - Contexto local, somente leitura e escrita
  - Controle dos níveis é feito através de travas

Concorrência híbrida baseada em threads

- Ambiente de programação baseado em threads
  - Em nível de usuário
- Trechos entre chamadas de sistema são convertidas em eventos atômicos
  - Compartilham um pool de thread de kernel cooperativamente
- Suporte a operações bloqueantes através de wrappers
  - Inversão de controle é escondida
- Escalonador executa no nível do usuário

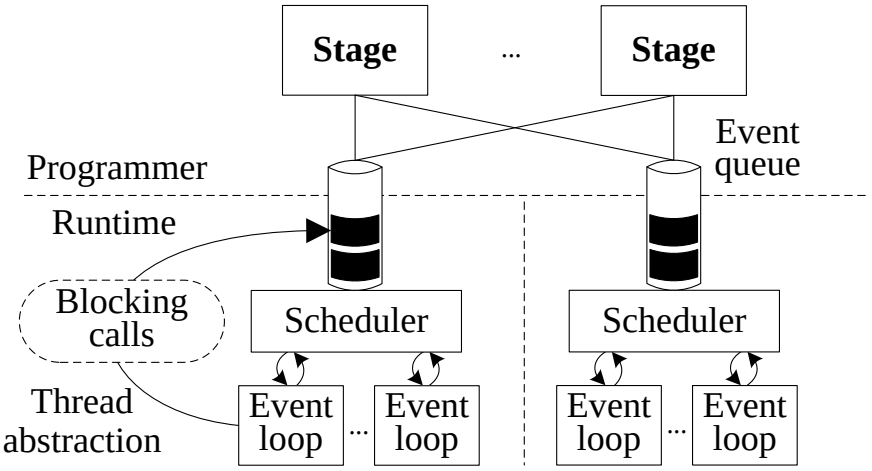


Threads híbridas: Algumas abordagens

- Capriccio
  - User threads em C (API Posix threads)
  - Corotinas para empacotar chamadas bloqueantes
  - Sincronização por bloqueio
  - Pool de threads de kernel pré-alocado e estático
- Scala Actors
  - Usa o modelo de atores para oferecer threads e eventos no mesmo ambiente
  - Atores compartilham um pool de threads dinâmico
  - Atores podem efetuar chamadas bloqueantes
  - Isolamento de memória
  - São especificados como threads
    - Conversão para o modelo de eventos via continuações
- Concorrência híbrida monádica
  - Abordagem funcional similar ao Capriccio
  - Usa o conceito de monads de continuação de Haskell para esconder a inversão de controle (continuações)
  - Sincronização por bloqueio

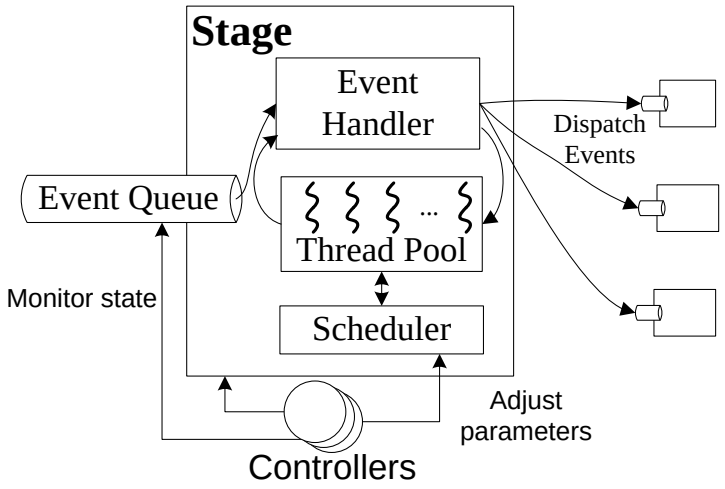
Concorrência em estágios

- Arquitetura para concorrência massiva
- Aplicação é dividida em estágios
  - Pipeline
  - Comunicam-se via fila de eventos
  - Cada estágio é auto-contido
  - Implementados com abstrações de threads
- Sincronização
  - Intra-estágio e inter-estágios



SEDA: Staged Event-Driven Architecture

- Tratador de eventos
  - Implementa a lógica do estágio
- Escalonador ciente de recursos
  - Controladores continuamente monitoram o estágio e ajustam parâmetros do escalonador
    - Tamanho do pool
  - Política por estágio
- Estágios podem bloquear
  - Usam múltiplas threads
- Sincronização dentro do estágio ainda é necessária



Referências

- Welsh, M., Culler, D., Brewer, E.: Seda: an architecture for well-conditioned, scalable internet services. (2001)
- Dabek, F., Zeldovich, N., Kaashoek, F., Mazieres, D., Morris, R.: Event-driven programming for robust software. (2002)
- Ururahy C., Rodriguez N., I.R.: Alua: flexibility for parallel programming. (2002)
- Behren, R.V., Condit, J., Zhou, F., Necula, G.C., Brewer, E.: Capriccio: Scalable threads for internet services. (2003)
- Haller, P., Odersky, M.: Actors that unify threads and events. (2007)
- Li, P., Zdancewic, S.: Combining events and threads for scalable network services implementation and evaluation of monadic, application-level concurrency primitives. (2007)
- Upadhyaya, G., Pai, V.S., Midkiff, S.P.: Expressing and exploiting concurrency in networked applications with aspen (2007)
- Han, B., Luan, Z., Zhu, D., Ren, Y., Chen, T., Wang, Y., Wu, Z.: An improved staged event driven architecture for Master-Worker network computing. (2009)
- Yoo, S., Lee, H., Killian, C., Kulkarni, M.: Incontext: simple parallelism for distributed applications. (2011)

Modelos de concorrência

Programação concorrente

Tiago Salmito

tsalmito@inf.puc-rio.br