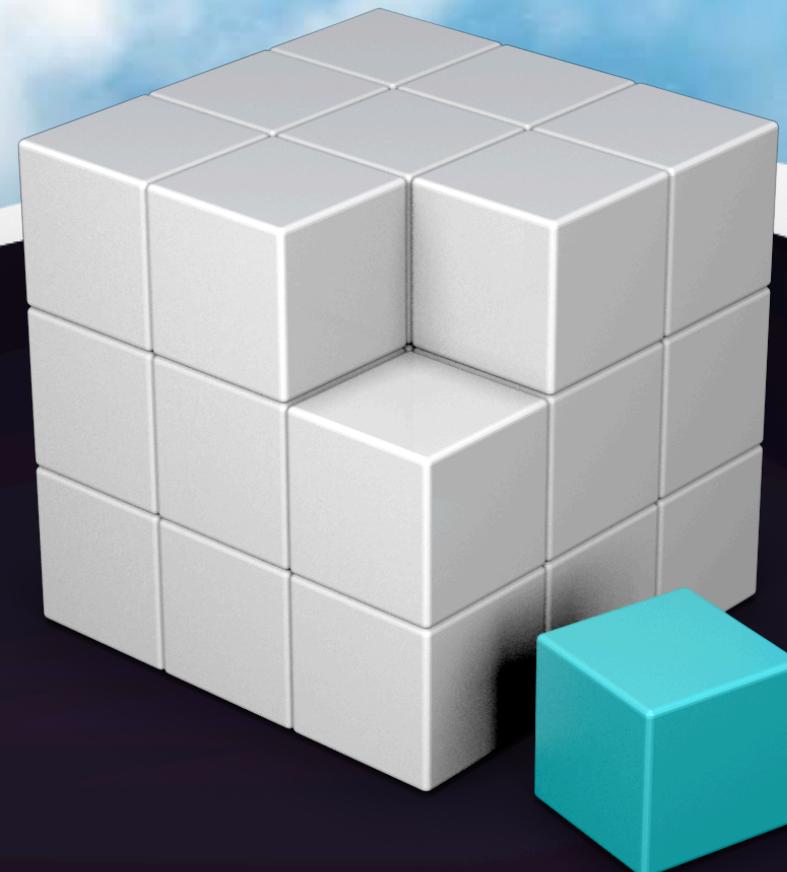


Avaliação e Desempenho de Sistemas Computacionais

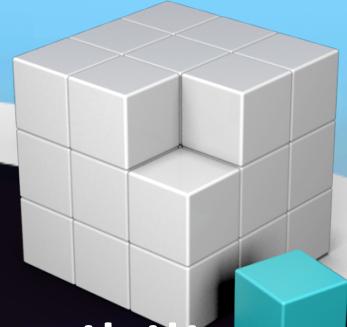
Técnicas e Classificação

UFPB / CI / DSC – 2019.1

Josilene Aires Moreira

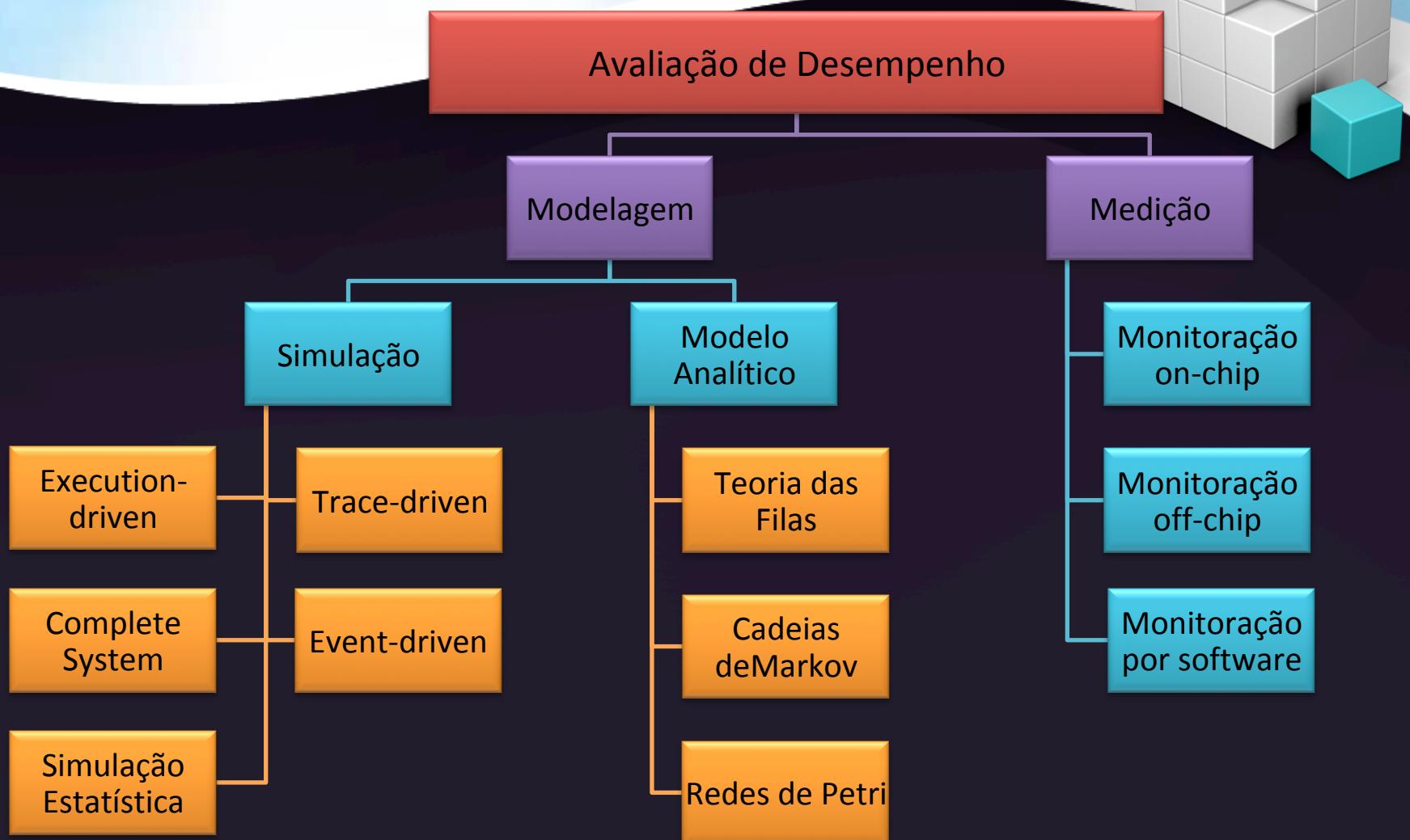


Avaliação de Desempenho



- Conjunto de técnicas e métodos que possibilitam investigar o comportamento temporal de sistemas
 - Tradição no estudo e dimensionamento de sistemas de comunicação, sistemas de manufatura, pesquisa operacional
 - Avaliação de desempenho de sistemas computacionais iniciou nos anos 60
 - Estudo sobre sistemas concorrentes (anos 60)
 - Sistemas de recursos compartilhados × Sistemas de tempo real

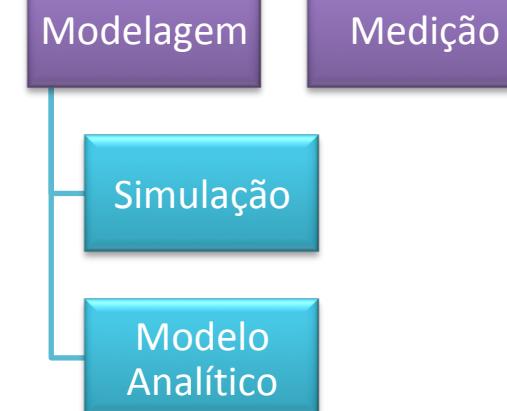
Avaliação de Desempenho



- Classificação de técnicas de avaliação de desempenho I (Eckhout, 2006)

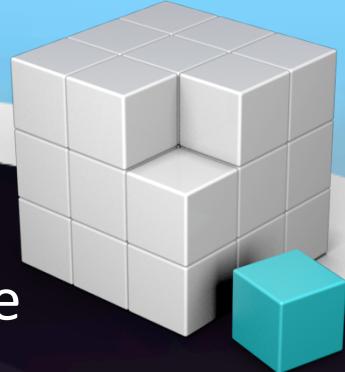
Avaliação de Desempenho

Avaliação de Desempenho

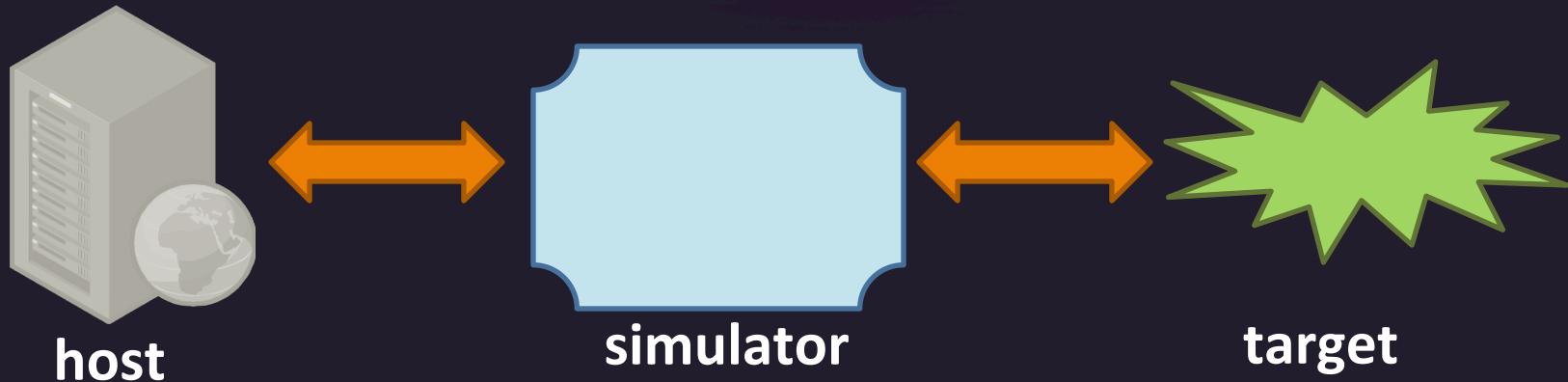


- Por que Modelar?
 - Medição só pode ser realizada se o sistema existir, ou sobre um protótipo
 - Sistemas a serem construídos podem ser estudados com modelagem
 - Modelagem através de Simulação ou Modelos Analíticos

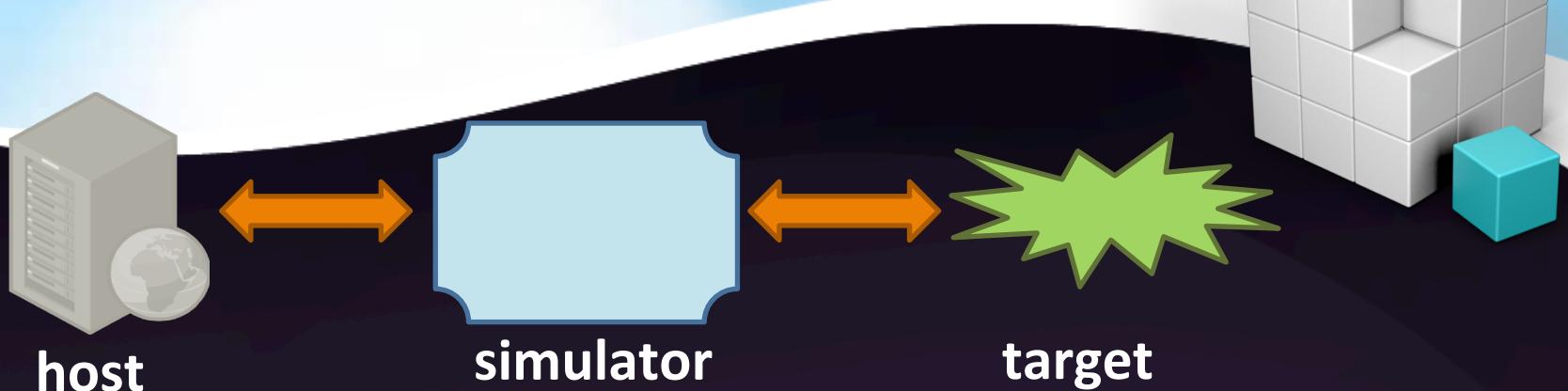
Simulação



- Método *de-facto* para a avaliação de desempenho de microprocessadores e arquiteturas de computador
 - Modelos analíticos podem não ser tão precisos
 - Simulação modela máquinas existentes ou máquinas futuras
- Simulador
 - É essencialmente um software que modela o sistema a ser simulado
 - Escrito em C ou Java
 - Rodando em uma máquina denominada host

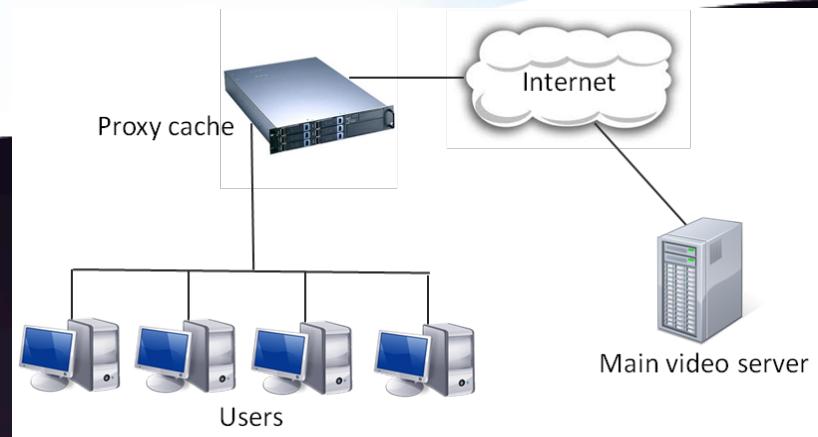


Simulador



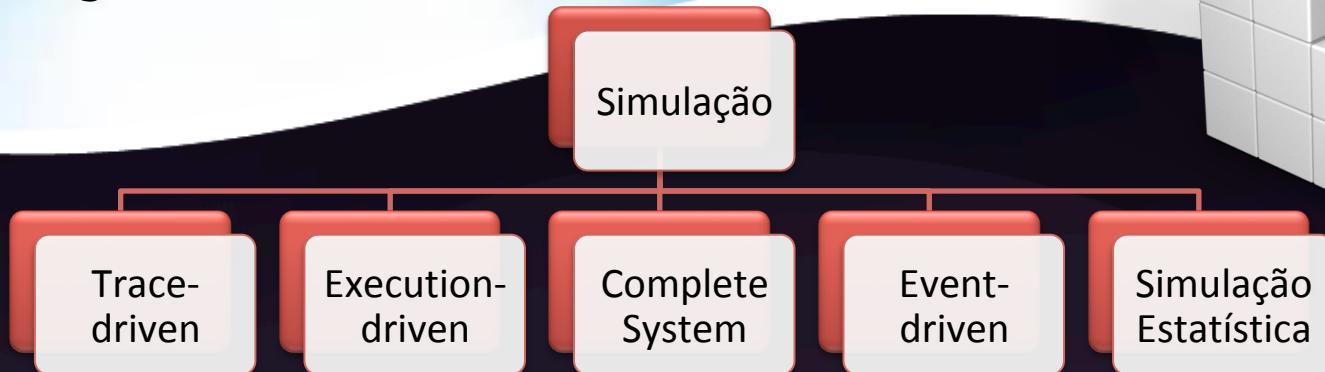
- Funcional
 - Provê essencialmente um componente muito similar ao que está sendo modelado
 - Pode ser modificados para prover informações de desempenho
 - Ex. Modelo de um microprocessador + informações de performance
- Temporal
 - Se a avaliação de desempenho é o único objetivo, não é necessário ter um modelo funcional
 - Não é necessário modelar todo o sistema
 - Incorpora as funcionalidades que se deseja avaliar

Simulador



- Um exemplo: simulando uma cache
 - Contém réplicas dos objetos, como vídeos
 - Diminui os atrasos e o congestionamento causados pela transferência dos objetos do servidor principal
 - Tem espaço finito
 - Objetos precisam ser armazenados
 - Como gerenciar os objetos que permanecem na cache?

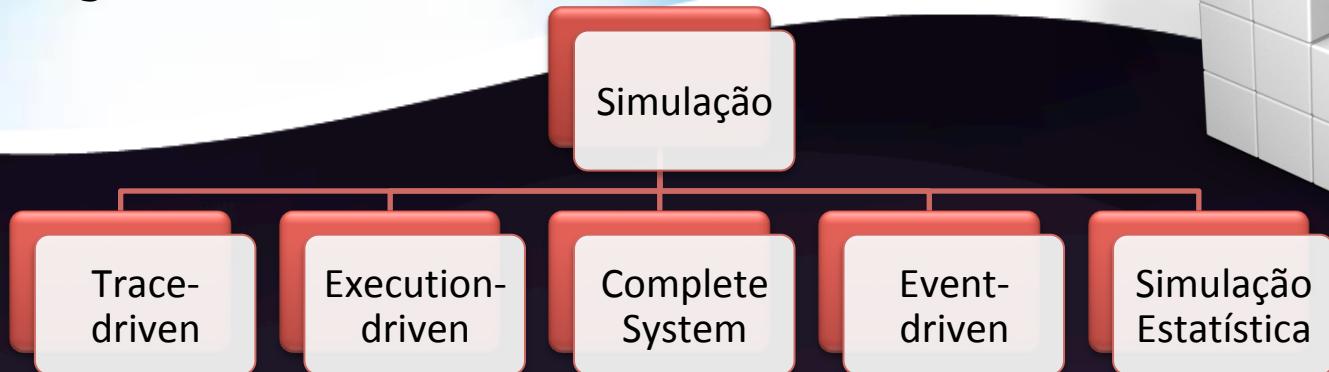
Simulação



- **Trace-driven Simulation**

- Consiste de um modelo de simulação onde as entradas são modeladas como um trace ou uma sequência de informações representando a sequência de instruções que seriam executadas na máquina target
- Um simulador de cache trace-driven precisaria, por exemplo, da sequência de requisições recebidas para os objetos do sistema
 - Se os dados (objetos) forem realmente movidos para dentro do espaço da cache = simulador funcional
 - Se apenas as referências (tamanho do objeto, posição do objeto) forem guardadas pelo simulador = simulador não-funcional (ou temporal)

Simulação



- Trace-driven Simulation

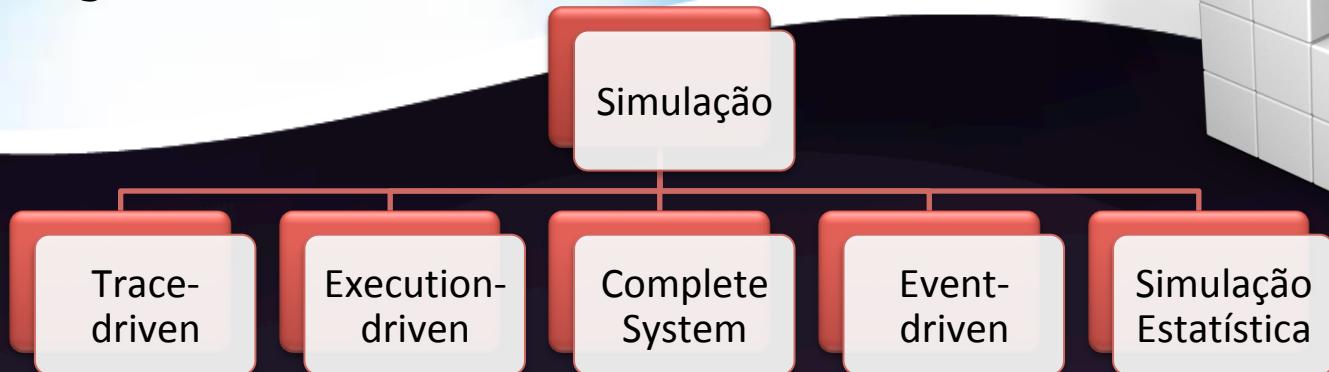
- Vantagens

- Simples de compreender
 - Fácil de *debugar*
 - Traces podem ser compartilhados entre pesquisadores

- Desvantagens

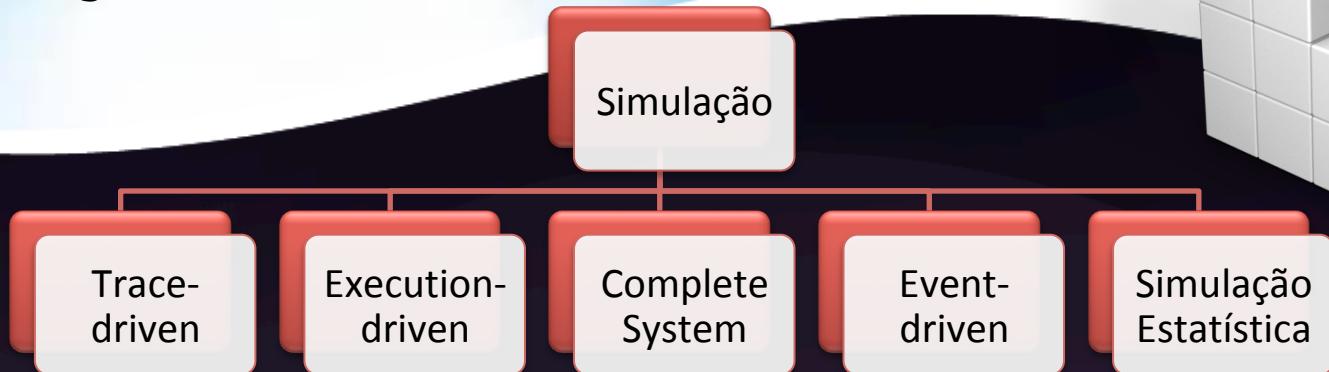
- Podem ser extremamente longos, se dados reais forem usados
 - Podem não ser muito representativos, pois um determinado trace pode não conter todas as situações possíveis

Simulação



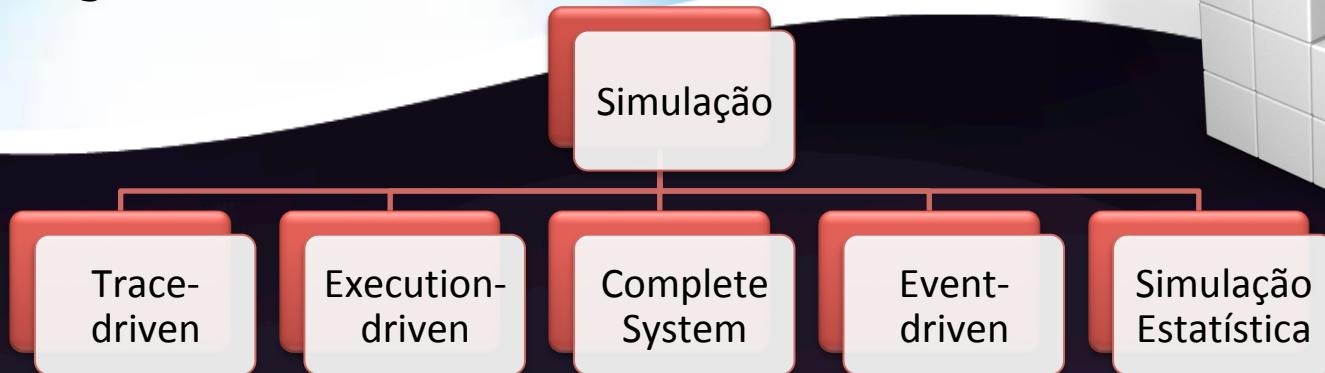
- Execution-driven Simulation
 - Específicos
 - Simuladores que recebem programas executáveis como entrada OU (nomenclatura)
 - Simuladores que se baseiam na execução de parte do código na máquina host – Não simulam os todas as instruções, mas apenas as que interessam

Simulação



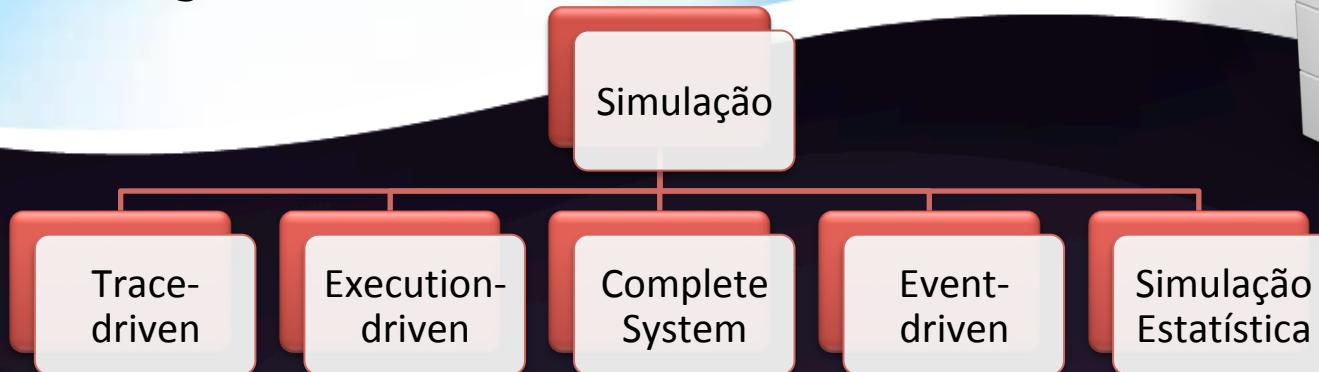
- Execution-driven Simulation
 - Vantagens
 - Muito precisos
 - Desvantagens
 - Consomem muito tempo de execução
 - Muitoooo tempo de desenvolvimento
 - Aplicação
 - Simulação de novas arquiteturas de processadores
 - Arquiteturas mudam frequentemente -> é desejável que os simuladores possam acompanhar a evolução

Simulação



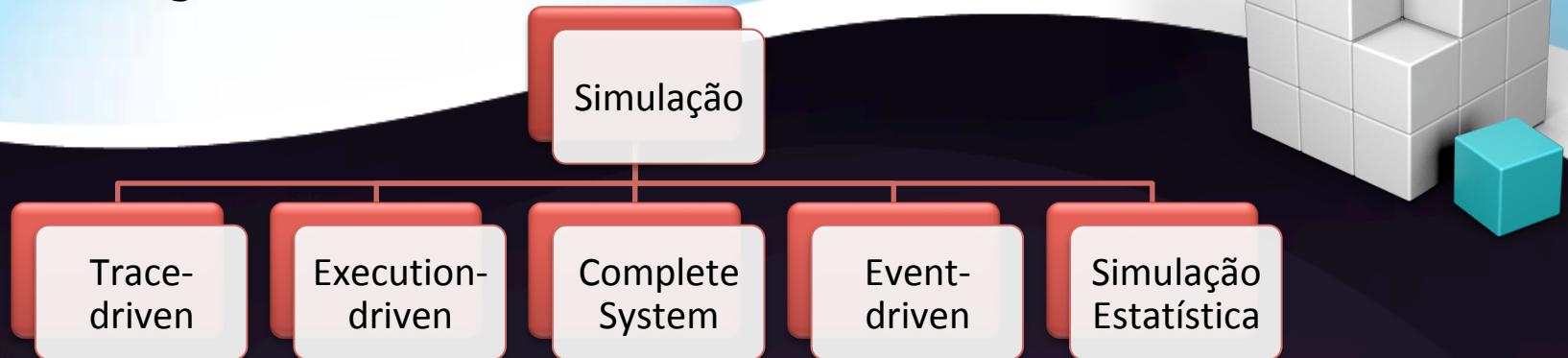
- Complete System Simulation
 - Grande parte das simulações não considera os tempos de I/O do sistema, nem o tempo das tarefas do SO
 - Complete system simulators são ambientes de simulação completos que modelam componentes de hardware com amplo nível de detalhe
 - processadores, memória, discos, controladores de rede, controladores gráficos, dispositivos seriais, timers, ...
 - Usados quando é importante simular o sistema completo

Simulação



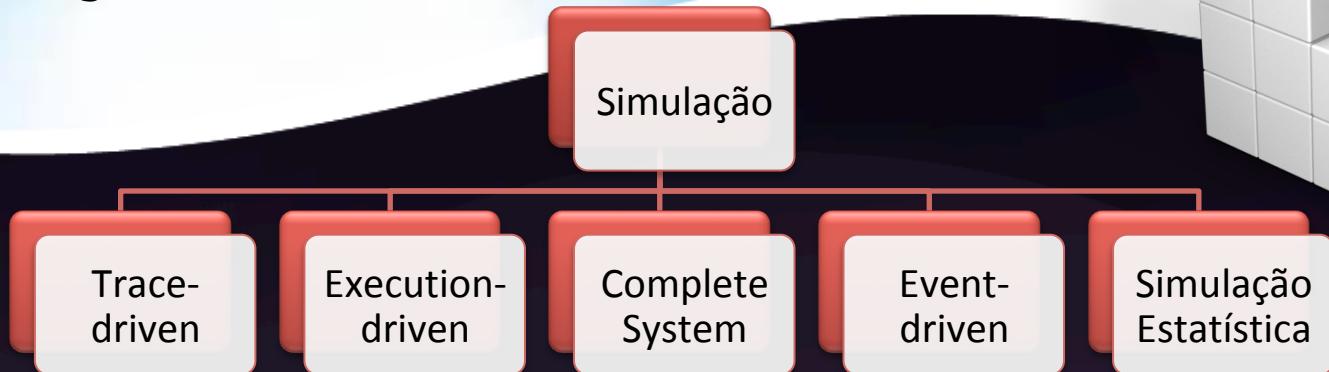
- Complete System Simulation
 - Vantagens
 - Todo o sistema pode ser simulado, incluindo o sistema operacional
 - Muito precisos
 - Desvantagens
 - Podem ser extremamente lentos
 - Difíceis de serem desenvolvidos

Simulação



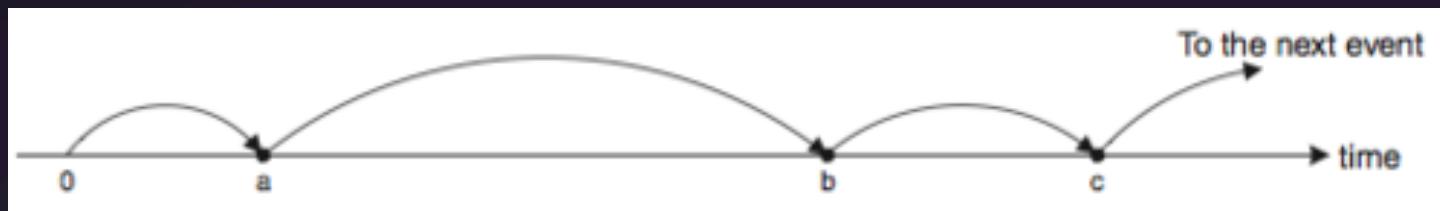
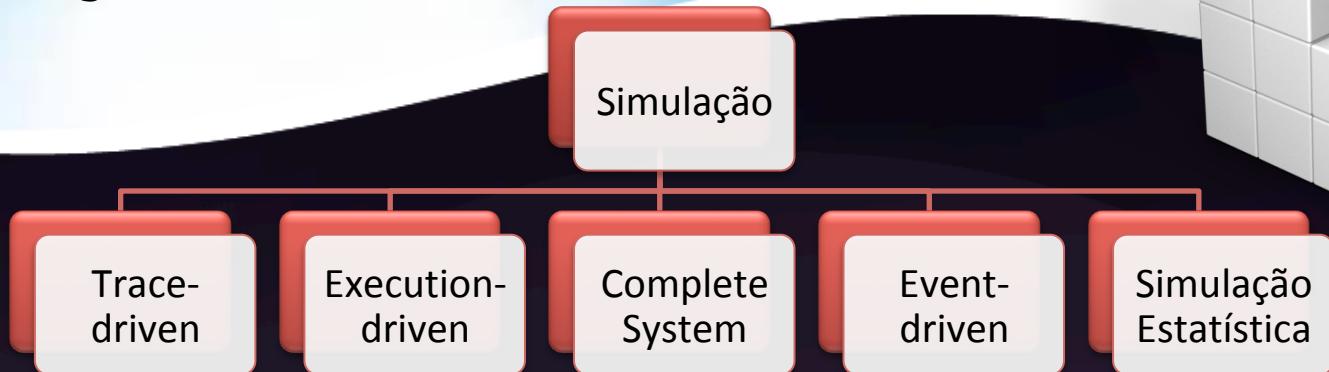
- Event-driven Simulation
 - Os simuladores anteriores basicamente realizavam as simulações ciclo-a-ciclo.
 - Uma simulação ciclo-a-ciclo imita a operação de um processador simulando as ações em cada ciclo
 - Em muitos ciclos, as diversas unidades do simulador permanecem sem tarefas a realizar
 - Essas operações reproduzem o comportamento de um processador ou computador, mas são extremamente lentas

Simulação



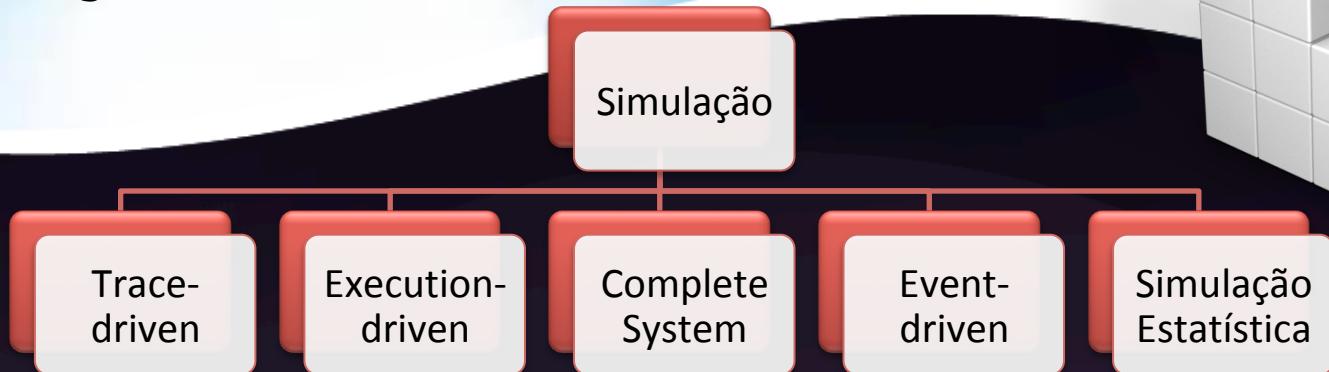
- Event-driven Simulation
 - Uma alternativa é criar um simulador onde eventos são escalonados para tempos específicos
 - O simulador “percebe” os eventos escalonados e realiza as tarefas escalonadas no tempo programado
 - As tarefas são colocadas em uma fila
 - O escalonador processa os eventos no tempo

Simulação



- O tempo de simulação inicia no primeiro evento (a)
- Executa eventos e move a execução para o próximo evento
- Não há noção de intervalo fixo de tempo
- Considera uma time line (linha do tempo) de simulação
- Não retrocede no tempo

Simulação



- Event-driven Simulation

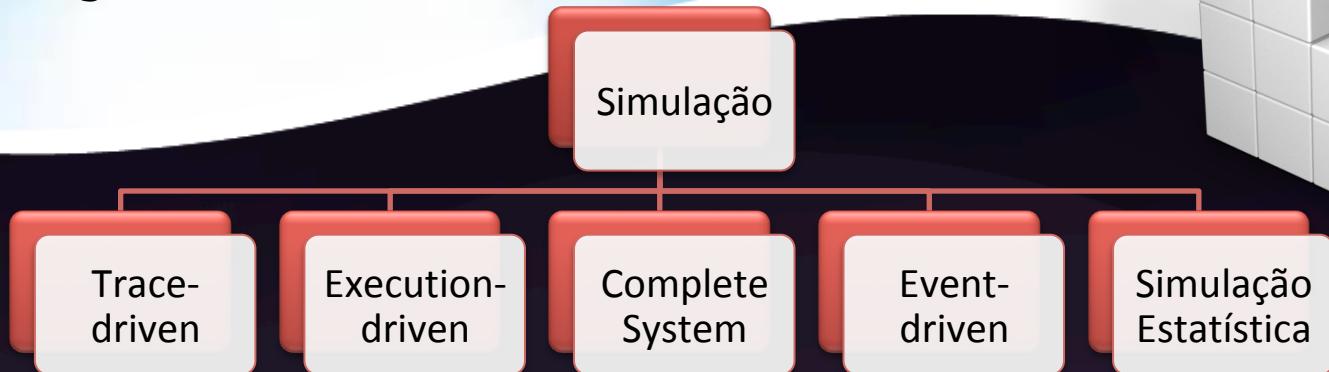
- Vantagens

- É utilizado em diversas áreas da análise de desempenho em computação=> Amplamente conhecido
 - Pode ser desenvolvido em linguagens de propósito geral, como C ou Java

- Observação

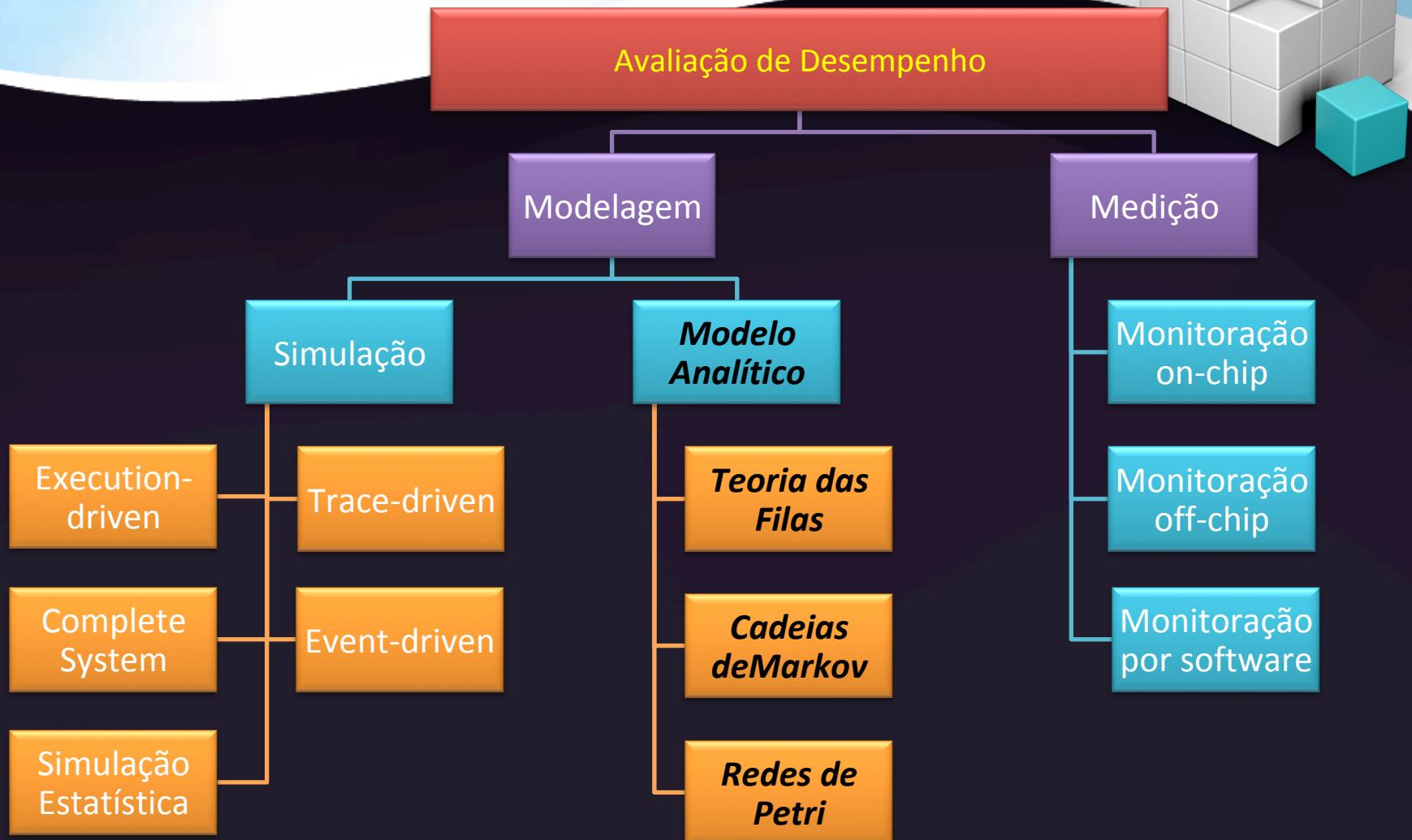
- Os dados de entrada podem vir de um trace, ou seguirem uma distribuição de probabilidade
 - Ex: As requisições de um simulador de cache para páginas Web seguem uma distribuição Zipf

Simulação



- Simulação Estatística
 - Técnica que usa traces gerados estatisticamente juntamente com um modelo onde os componentes são modelados também estatisticamente
 - Muito usado para modelagem de processadores
 - Envolve conhecimento de modelagem estatística

Avaliação de Desempenho



- Classificação de técnicas de avaliação de desempenho I (Eckhout, 2006)

Avaliação de Desempenho

Modelagem

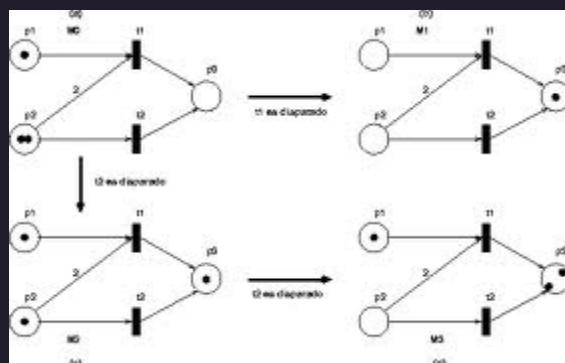
Modelo
Analítico

Teoria das
Filas

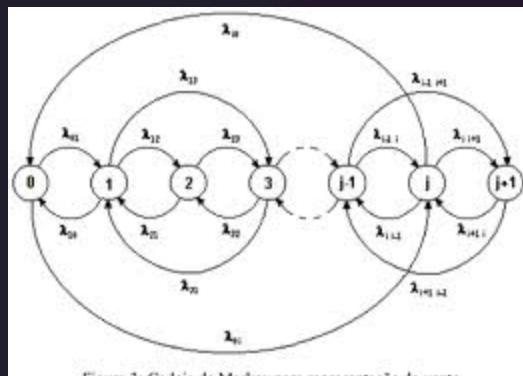
Cadeias
de Markov

Redes de
Petri

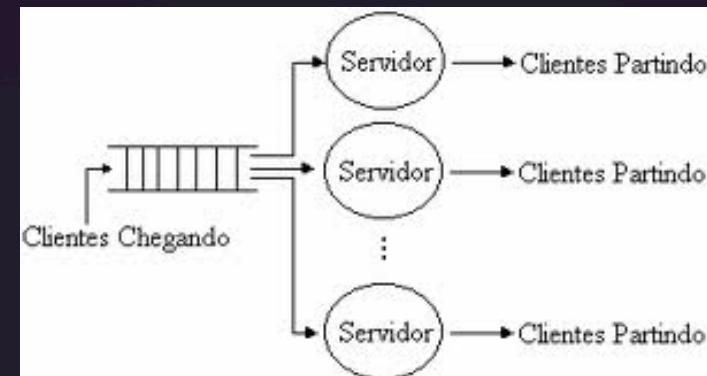
- Modelos analíticos
 - Baseiam-se em métodos probabilísticos para criar modelos dos sistemas computacionais
 - teoria das filas
 - cadeias de Markov
 - redes de Petri
 - Existem desde os anos 70



Redes de Petri



Cadeias de Markov



Teoria das Filas

Avaliação de Desempenho

- Modelos analíticos
 - Vantagens
 - Utilizam equações matemáticas
 - São considerados de baixo custo
 - Desvantagens
 - Para obterem problemas tratáveis, isto é, com solução, precisam simplificar o problema
 - Assim, não são tão precisos
 - Observações
 - Utilizado para planejamento de capacidade
 - Não muito usado para avaliação de desempenho de processadores devido à falta de precisão
- As técnicas de Simulação Estatística
 - Podem ser consideradas como técnicas híbridas
 - Junção de Simulação com Modelagem Analítica



Avaliação de Desempenho



- Classificação de técnicas de avaliação de desempenho I (Eckhout, 2006)

Avaliação de Desempenho

Medição

Monitoração
on-chip

Monitoração
off-chip

Monitoração
por software

- Performance measurement (Medição)
 - Usado em sistemas prontos ou protótipos
 - Ajuste fino (tunning)
 - Validação de desempenho
 - Testes de performance com diferentes aplicações
 - Envolve
 - Descobrir os gargalos do sistema
 - Compreender os tipos de aplicação e como são afetadas pelos gargalos
 - Projetar novas características do sistema

Avaliação de Desempenho

Medição

Monitoração
on-chip

Monitoração
off-chip

Monitoração
por software

- Performance measurement (Medição)
 - Muitos sistemas são construídos com características configuráveis
 - Ex. Registradores em um processador podem ou não realizar *prefetching*
 - Como o sistema se comporta ao se modificar uma determinada característica?
 - Medição pode revelar informações críticas sobre a eficiência dos microprocessadores sob diferentes características e diferentes cargas de dados (*workloads*)



Avaliação de Desempenho

Medição

Monitoração
on-chip

Monitoração
off-chip

Monitoração
por software

- Medição *On-chip*

- Microprocessadores mais modernos (Intel, IBM, AMD, Sun,...) incorporam monitores de performance no chip
- Usados para compreender a performance dos processadores em diferentes / complexas situações, sob distintas cargas de dados (*workloads*)
- Esta habilidade supera as limitações dos simuladores



Avaliação de Desempenho

Medição

Monitoração
on-chip

Monitoração
off-chip

Monitoração
por software

- Medição *On-chip* (ex.)
 - Contagem de ciclos de execução
 - Contagem de acertos / erros na memória cache
 - Contadores
 - Atividades do kernel do processador
 - Atividades dos usuários
 - Tools
 - Vtun (<http://software.intel.com/en-us/intel-vtune-amplifier-xe>)



Avaliação de Desempenho

Medição

Monitoração
on-chip

Monitoração
off-chip

Monitoração
por software

- Medição *Off-chip*

- Quando se acrescenta hardware para avaliar o desempenho dos processadores
- São usados “tracers” que interrompem a execução dos programas a cada instrução
- O estado da CPU é coletado
- Os dados são armazenados para análise posterior
- O sistema operacional também pode ser avaliado



Medição

Monitoração
on-chip

Monitoração
off-chip

Monitoração
por software

Avaliação de Desempenho

- Monitoração por software
 - São introduzidos breakpoints nos sistemas reais ou nos protótipos
 - Tem a vantagem de ser fácil de implementar
 - Pode tornar o sistema lento
 - Só é capaz de monitorar as atividades dos usuários
 - Dificilmente é possível usar para monitorar a atividade do SO

The image shows a debugger interface with two panes. The left pane displays assembly code with several red circular markers indicating breakpoints. The right pane shows Java code for a Factorial class. Red arrows point from the text "Click the margin to set Breakpoint" to the margin of the Java code editor, where two small circular icons are visible, one with a red border and one with a red dot.

```
/* setup APMC */
APMC_PCER = (1<<PIOA_ID) | (1<<PIOB_ID) | (1<<ADCO_ID) | (1<<USO_ID) | (1<<TCO_ID);

init_serial ();

/* setup A/D converter */
ADCO_CR = ADC_SWRST;
ADCO_CHER = ADC_CHO | ADC_CH1 | ADC_CH2;
ADCO_MR = ADC_8_BIT_RES | (1<<ADC_BI);

public class Factorial {
    // Print factorial of n
    public static void main(String[] args) {
        int n = 20;
        int factorial = 1;

        // n! = 1*2*3...
        for (int i = 1; i <= n; i++) {
            factorial *= i;
        }
        System.out.println("The Factorial of " + n + " is " + factorial);
    }
}
```

Medição

Monitoração
on-chip

Monitoração
off-chip

Monitoração
por software

Avaliação de Desempenho

- Monitoração por software
 - São introduzidos breakpoints nos sistemas reais ou nos protótipos
 - Tem a vantagem de ser fácil de implementar
 - Pode tornar o sistema lento
 - Só é capaz de monitorar as atividades dos usuários
 - Dificilmente é possível usar para monitorar a atividade do SO

The image shows a debugger interface with two panes. The left pane displays assembly code with several red circular markers indicating breakpoints. The right pane shows Java code for a Factorial class. Red arrows point from the text "Click the margin to set Breakpoint" to the margin of the Java code editor, where two small circular icons are visible, one with a red border and one with a red dot.

```
/* setup APMC */
APMC_PCER = (1<<PIOA_ID) | (1<<PIOB_ID) | (1<<ADCO_ID) | (1<<USO_ID) | (1<<TCO_ID);

init_serial ();

/* setup A/D converter */
ADCO_CR = ADC_SWRST;
ADCO_CHER = ADC_CHO | ADC_CH1 | ADC_CH2;
ADCO_MR = ADC_8_BIT_RES | (1<<ADC_BI);

public class Factorial {
    // Print factorial of n
    public static void main(String[] args) {
        int n = 20;
        int factorial = 1;

        // n! = 1*2*3...
        for (int i = 1; i <= n; i++) {
            factorial *= i;
        }
        System.out.println("The Factorial of " + n + " is " + factorial);
    }
}
```

Avaliação de Desempenho

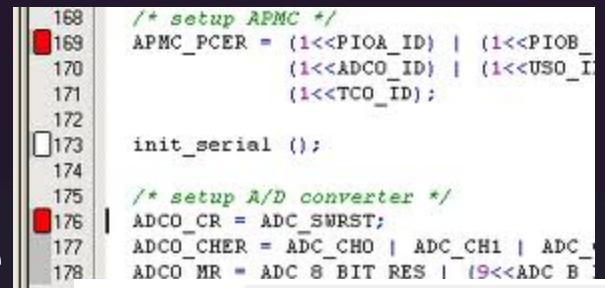
Medição

Monitoração
on-chip

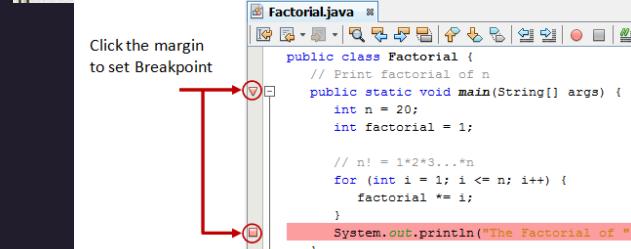
Monitoração
off-chip

Monitoração
por software

- Consumo de Energia
 - Atualmente, tornou-se importante avaliar o consumo de energia dos processadores
 - Construção de modelos dos diversos módulos contidos na arquitetura do microprocessador, para medir o consumo das operações, tipicamente
 - Leitura e gravação nos registradores
 - Acesso à memória cache
 - Acesso aos barramentos



```
168 /* setup APMC */
169 APMC_PCER = (1<<PIOA_ID) | (1<<PIOB_
170 (1<<ADCO_ID) | (1<<USO_I
171 (1<<TCO_ID);
172
173
174
175
176 ADCO_CR = ADC_SWRST;
177 ADCO_CHER = ADC_CHO | ADC_CH1 | ADC_
178 ADCO_MR = ADC_8_BIT_RES | (9<<ADC_B
```



Click the margin to set Breakpoint

```
Factorial.java */
public class Factorial {
    // Print factorial of n
    public static void main(String[] args) {
        int n = 20;
        int factorial = 1;

        // n! = 1*2*3...*n
        for (int i = 1; i <= n; i++) {
            factorial *= i;
        }
        System.out.println("The Factorial of " + factorial);
    }
}
```