Estimação do WCET usando Análise Estática

Fundamentos dos Sistemas de Tempo Real

Rômulo Silva de Oliveira

eBook Kindle, 2018

www.romulosilvadeoliveira.eng.br/livrotemporeal

Outubro/2018

Sumário

- Introdução
- Importância do Tipo de Hardware
- Análise Estática do WCET
- Análise do Fluxo de Controle
- Análise do Comportamento do Processador
- Análise da Memória Cache
- Análise do Pipeline
- Cálculo do Limite Superior para o WCET
- A Ferramenta aiT
- Considerações Finais

Introdução 1/10

- Sistemas de tempo real são classificados de acordo com a criticalidade dos requisitos temporais
  - Sistemas críticos ou hard real-time
  - Sistemas não críticos ou soft real-time

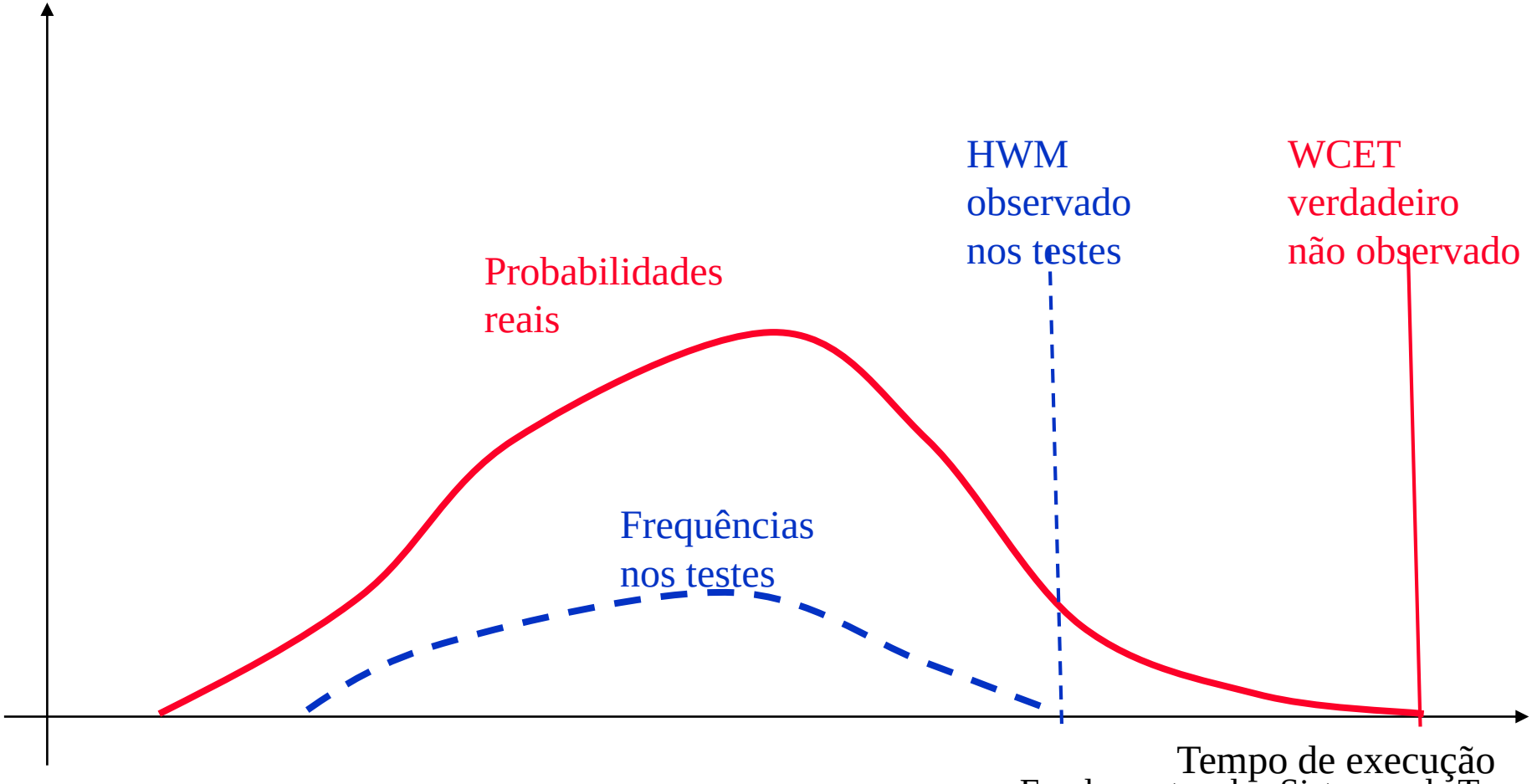
- Para sistemas críticos é necessário oferecer garantias quanto ao atendimento dos prazos
- É necessário conhecer o tempo de execução no pior caso de cada tarefa (WCET – Worst Case Execution Time)
- Mesmo para sistemas não críticos é uma informação relevante

Introdução 2/10

- Existe uma demanda crescente por capacidade de processamento
  - Também nos sistemas de tempo real
- Maior capacidade de processamento requer o uso de arquiteturas de computador modernas e complexas
- Arquiteturas modernas são projetadas visando a redução do tempo médio de execução (Average-Case Execution Time – ACET)
- Arquiteturas modernas:
  - Geram tempos de computação variáveis
  - Podem apresentar um comportamento patológico no pior caso
  - Pior caso tem probabilidade muito pequena, porém maior que zero

Introdução 3/10

- Distribuição dos tempos de execução de uma tarefa
  - Probabilidades considerando todos os cenários e entradas possíveis P(c)
  - Frequências observadas durante um certo número de testes F(c)



Introdução 4/10

- O tempo de execução no pior caso supõe os piores comportamentos para o software e para o hardware

- O pior fluxo de controle para cada tarefa
  - Os piores dados de entrada
  - Os piores valores para variáveis globais
- O pior comportamento das caches
- O pior comportamento do pipeline
- O pior comportamento do branch predictor
- Etc

- O pior comportamento de tudo

Introdução 5/10

- O WCET é determinado por múltiplos fatores cuja análise criteriosa é fundamental para a determinação de seu valor
  - Ou uma estimativa que represente um limite superior seguro (*upper-bound*)

- A precisão da estimativa do WCET vai depender das técnicas de análise utilizadas

- Análise estática
  - Analisa o código sem executa-lo
- Medições
- Métodos híbridos misturando os dois anteriores

Introdução 6/10

- Não é possível obter limites superiores para os tempos de execução de programas em geral
  - Seria equivalente a resolver o halting problem
- Para determinar o WCET é necessário uma forma restrita de programação, a qual garante que o programa sempre termina
  - Recursão não é permitida
  - Laços possuem um número de iterações explicitamente limitado
- Em geral, o cenário que gera o pior tempo de execução de uma tarefa não é conhecido de antemão, e muito difícil de identificar
  - Dados de entrada
  - Valores das variáveis globais
  - Estado inicial do hardware (cache, pipeline, branch predictor, etc)

Introdução 7/10

- O método mais empregado na prática para estimar o tempo de execução no pior caso é simplesmente executar a tarefa
  - Mede o tempo de execução da tarefa sozinha
  - Para um certo número de cenários (os casos de teste)

- Com isto é obtido o máximo tempo de execução observado
  - Linha azul na figura
  - Também chamado de HWM (High Water Mark)

- Normalmente o máximo tempo de execução observado será menor que o verdadeiro (e desconhecido) WCET
  - Linha vermelha na figura
  - Na prática pode-se adicionar uma margem de segurança (20% por exemplo)

Introdução 8/10

- Algumas abordagens medem o tempo de execução de diferentes partes do código, e não da tarefa inteira
- Estas medições do tempo de execução de segmentos são depois combinadas, da pior forma possível, para obter-se uma estimativa mais segura para o WCET

- Mesmo assim, não existe garantia de que a estimativa obtida é maior que o verdadeiro WCET

Introdução 9/10

- Um estimativa garantidamente segura para o WCET requer que o método usado considere todos os possíveis tempos de execução

- Tais métodos realizam uma análise estática da tarefa, sem executa-la realmente
  - Uma visão abstrata da tarefa é empregada para simplificar o trabalho

- Esta visão abstrata da tarefa implica alguma perda de informação
  - Uma simplificação pessimista da tarefa é empregada
  - A estimativa obtida é um limite superior para WCET (upper bound)
  - A estimativa obtida é garantidamente maior que o verdadeiro WCET

- Exatamente o quanto pessimista é a estimativa depende dos métodos usados para fazer a análise estática e da previsibilidade do sistema em questão
  - Previsibilidade do comportamento do hardware
  - Previsibilidade do fluxo de controle da tarefa

Introdução 10/10

- Ao usar uma estimativa do WCET, algumas questões são relevantes

- A estimativa representa um valor aproximado do WCET ?
  - Para mais ou para menos
- Ou a estimativa representa um limite superior para o WCET ?
  - Sempre pessimista, sempre maior que o verdadeiro WCET

- Uma informação relevante, mas muito difícil de ser obtida, é:
  O quão próximo do verdadeiro WCET está a estimativa obtida ?

Importância do Tipo de Hardware 1/4

- O hardware usado tem grande impacto sobre a variância do tempo de execução e sua previsibilidade
- Caso mais fácil são processadores clássicos de 8 e 16 bits com arquiteturas simples
  - Cada instrução tem um tempo fixo de execução
- Tais processadores são fáceis de modelar com respeito aos tempos do hardware
- O principal trabalho a fazer na análise do WCET é determinar o grafo de fluxo de controle e o caminho mais longo
- Estes processadores são cada vez menos usados, em função da sua limitação de desempenho
  - Por exemplo, Intel MCS-51

Importância do Tipo de Hardware 2/4

- Existem também processadores com pipeline simples que executam as instruções exatamente na ordem que aparecem no programa
  - In-order pipeline
- São muito usados atualmente em computadores embutidos em equipamentos (embedded systems) com restrições de custo porém necessitando mais desempenho do que o disponível em processadores clássicos de 8 e 16 bits
  - Por exemplo, ARM7 e ARM Cortex R series

- Incluem pipeline e memória cache, porém com uma solução simples
- É possível fazer a análise estática do WCET neste tipo de hardware

Importância do Tipo de Hardware 3/4

- Para aplicações mais complexas, que exigem maior desempenho do hardware, é necessário empregar processadores com arquiteturas complexas

- Nestes processadores existem memórias cache, pipelines com execução fora de ordem
  - Por exemplo, PowerPC 750, PowerPC 7448 e ARM11

- Análise estática deste tipo de processador é muito mais difícil

Importância do Tipo de Hardware 4/4

- A tendência é a arquitetura dos computadores ficar sempre mais complexa
  - Elementos especulativos (com comportamento probabilista) são usados para aumentar o desempenho médio

- Tais arquiteturas utilizam
  - Múltiplas threads de hardware em cada core do processador
  - Pipelines com muitos estágios em paralelo (deep pipelines)
  - Diversos níveis de cache

- Por exemplo, AMD Opteron, Intel Core i7, IBM Power8

Análise Estática do WCET 1/4

- Objetivo é determinar o pior caso do tempo de execução
  - Sem realmente executar a tarefa
- As técnicas de análise estática determinam estimativas através de modelos do programa e do hardware
  - Os modelos devem ser corretos, mas não são necessariamente exatos
- Os valores obtidos são seguros
  - O WCET estimado sempre será maior do que o WCET real
- Na falta de informações do modelo
  - Tempo de execução pode ser extremamente pessimista

- Na prática: obtenção de um limite superior (*upper bound*)
- A análise estática é discutida em detalhes no artigo ACM Transactions on Embedded Computing Systems, Vol. 7, No. 3, Article 36

Análise Estática do WCET 2/4

- Análise estática determina um limite superior para o tempo de execução de uma **tarefa** quando executada em um **hardware específico**
- O tempo de uma execução em particular depende
  - Do caminho na tarefa feito pelo fluxo de controle
  - Do tempo gasto nas instruções de máquina neste caminho e neste hardware

- A determinação de um limite para o tempo de execução precisa considerar
  - Todos os caminhos possíveis para o fluxo de controle
  - Os tempos de execução das instruções neste conjunto de caminhos

- Abordagem típica divide o problema da análise em uma sequência de subproblemas
  - Alguns lidam com as questões do fluxo de controle
  - Outros lidam com o tempo de execução de sequências de instruções de máquina

- Vários métodos são usados para atacar os vários subproblemas

Análise Estática do WCET 3/4

- A análise estática apresenta várias limitações
- O número máximo de iterações em laços precisa ser conhecido
- Na maioria das vezes recursão é proibida
- Não é possível usar ponteiros para dados e funções que não possam ser resolvidos (conhecidos) estaticamente
- Muitas vezes não é possível usar alocação dinâmica de memória
- Muitas análises requerem que o padrão para chamada de função no processador em questão seja rigorosamente seguido
- Atualmente apenas monoprocessadores são suportados
- A maioria das análises supõe que a tarefa analisada seja executada sem interrupções
  - Interrupções desabilitadas

Análise Estática do WCET 4/4

- Determinar o número máximo de iterações para cada laço da tarefa pode ser difícil
- Modelar corretamente o comportamento do processador, com seus diversos elementos, pode ser muito difícil
- A preempção da tarefa por tratadores de interrupção ou outras tarefas de mais alta prioridade causam grandes pertubações no contexto de execução
- A execução de uma tarefa de mais alta prioridade vai arruinar o conteúdo da cache para a tarefa que sofreu a preempção
  - Ao voltar a tarefa preemptada precisa recarregar a memória cache
- A evolução da arquitetura dos computadores torna a análise estática dos tempos de execução cada vez mais difícil

Análise do Fluxo de Controle 1/10

- Uma tarefa exibe seu WCET em um caminho de execução específico
  - WCEP (Worst-Case ExecutionPath)
  - Algumas vezes diversos caminhos podem levar ao WCET
- Se os dados de entrada e o estado inicial do hardware (cache, pipeline, branch-predictor, etc) que leva ao WCET fosse conhecido
  - Obter o WCET seria bem mais fácil
  - Bastaria executar a tarefa com estes dados de entrada a partir destas condições iniciais e medir o tempo de execução
- Em geral, isto não é possível

- O **Grafo de Fluxo de Controle** (**GFC**) é usado para descrever todos os caminhos de execução possíveis
  - CFG (Control-Flow Graph)
  - Inclui o Grafo de Chamadas que mostra qual subrotina chama qual

Análise do Fluxo de Controle 2/10

- Seria mais fácil construir o Grafo de Fluxo de Controle a partir do código fonte
- Mas é difícil mapear os resultados para o programa em código de máquina
  - Otimizações do compilador
  - Ligações com bibliotecas e entre módulos
- O Grafo de Fluxo de Controle é então construído a partir do código de máquina

- O GFC deve conter todas as instruções da tarefa

- Dificuldades são criadas por saltos dinâmicos (indexados)
  - Saltos dinâmicos surgem na implementação de comandos tipo switch/case
  - Pode requerer a análise do código fonte para identificar os alvos dos saltos

- Dificuldades são criadas por chamadas de subrotina com endereço cálculado
  - Por exemplo quando são usados ponteiros para função em C

Análise do Fluxo de Controle 3/10

- O propósito da análise do fluxo de controle é gerar informações sobre os possíveis caminhos da execução

- O conjunto de caminhos é sempre finito, pois a terminação do programa precisa ser garantida
  - O número exato de caminhos pode ser difícil de determinar
  - Mas uma aproximação segura pode ser usada

- Além do GFC, podem ser usados:
  - Limites para os valores possíveis dos dados de entrada
  - Limites para o número de iterações dos laços
  - Determinados por análise de valor ou fornecidos pelo desenvolvedor

Análise do Fluxo de Controle 4/10

- A anotação do código da tarefa com informações disponíveis do desenvolvedor da aplicação pode facilitar muito a análise
- O desenvolvedor pode suprir estas informações adicionais em arquivos separados ou como comentários no código fonte
- Informações tais como:
  - Layout de memória e características das diferentes áreas de memória
  - Valores possíveis para os dados de entrada
  - Limites para o número de iterações dos laços
  - Relações entre o número de iterações de laços internos e variáveis usados em laço externo
  - Frequência esperada para cada caminho em um saldo condicional
  - Chamadas de subrotina que fogem do padrão para o processador em questão

Análise do Fluxo de Controle 5/10

- Os diferentes possíveis caminhos da execução de uma tarefa são tomados ou não com base nos dados de entrada

- Alguns caminhos que aparecem no GFC podem ser impossíveis, não importando os dados de entrada
  - Por exemplo,
    IF(A>B) X();
    IF(A<B) Y();
    se X() não altera nem A nem B, jamais X() e Y() serão executados juntos

- Eliminar tais caminhos impossíveis vai aumentar a precisão da análise
  - O GFC pode mostrar como pior caminho na verdade um caminho impossível

Análise do Fluxo de Controle 6/10

- A análise do fluxo de controle busca eliminar caminhos impossíveis
- A análise do fluxo de controle também busca determinar um limite máximo para o número de iterações de cada laço
- Podem ser usadas informações adicionais fornecidas pelo programador (anotações no código fonte)
- São analisados os valores possíveis para as variáveis e as expressões dos comandos que podem causar desvios (Análise de Valor)

Análise do Fluxo de Controle 7/10

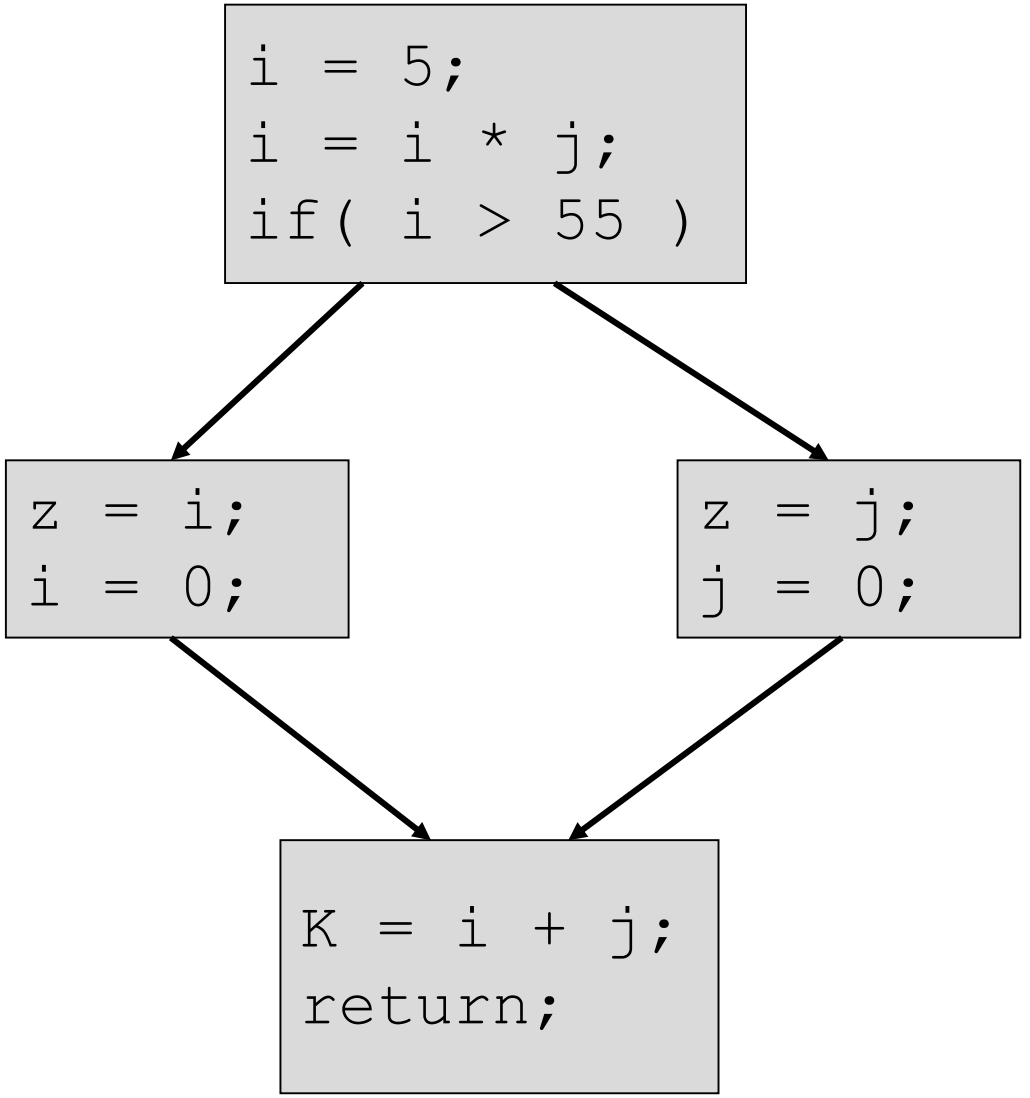
- **Análise de Valor** é capaz de determinar várias informações relevantes para a análise estática do tempo de execução
  - Especialmente em programas com código bem comportado
- A análise de valor computa os intervalos possíveis para valores armazenados nos registradores do processador, assim como variáveis de memória
- Ela permite determinar
  - Limites para o número de iterações de laços
  - Caminhos impossíveis
  - Endereços de memória efetivos para usar na análise da cache

Análise do Fluxo de Controle 8/10

- Muitas vezes os nodos do Grafo de Fluxo de Controle não são instruções de máquina individuais, mas sim blocos básicos
- Um Bloco Básico (BB) é uma sequência de instruções de máquina consecutivas nas quais a execução sempre inicia pela primeira instrução do bloco básico e sempre termina na última instrução do bloco básico
  - Não existem desvios para entrar no meio do bloco básico
  - Não existem desvios para sair no meio do bloco básico
- Em um bloco básico, o fluxo de execução sempre vai do início até o fim, sem interrupções nem desvios de fluxo

Análise do Fluxo de Controle 9/10

```
i = 5;
i = i * j;
if( i > 55 )
   {   z = i;
       i = 0;
       }
else
   {   z = j;
       j = 0;
       }
K = i + j;
return;
```



Análise do Fluxo de Controle 10/10

- Processadores complexos podem na verdade executar as instruções de máquina em uma ordem diferente daquela que aparece no código binário do programa
  - Ou seja, daquela suposta pela análise do fluxo de controle

- Isto acontece devido a técnicas tais como
  - Pipeline com busca antecipada (*prefetching*) e postergação de salto (*delayed branching*)
  - Previsão de saltos (*branch prediction*)
  - Execução especulativa ou fora de ordem

- Análise estática neste caso muito mais difícil ou mesmo impraticável

Análise do Comportamento do Processador 1/10

- Em processadores clássicos simples (8 e 16 bits) é possível assumir que o tempo de execução de uma instrução de máquina é constante
  - Ele independente de contexto
- O tempo de execução de uma dada instrução de máquina individual independe do que foi executado antes
  - Independe da história de execução
  - Estes tempos podem ser encontrados no manual do processador
  - Podem variar conforme tipo de operando ou area de memória acessada
- Neste caso,
  - Se a tarefa primeiro executa um trecho de código “A” e depois um trecho “B”
  - E o tempo de execução de “A” no pior caso é WCET(A)
  - E o tempo de execução de “B” no pior caso é WCET(B)
  - Então a execução no pior caso de “A;B” será WCET(A)+WCET(B)

Análise do Comportamento do Processador 2/10

- Esta independência de contexto não é verdadeira para processadores um pouco mais modernos (em geral 16 e 32 bits)
  - Com cache e pipeline simples (escalar)
- O tempo de execução de uma instrução de máquina individual pode variar
- Instruções de máquina apresentam tempo de execução dependente do contexto deixado pela história da execução até o momento
- No caso do exemplo anterior “A;B”, o tempo de execução de B depende do contexto deixado pela execução de “A”
  - Do estado deixado por “A” no pipeline, cache, etc
  - Podem existir vários caminhos para chegar até “B”, via saltos para “B”

Análise do Comportamento do Processador 3/10

- Limites para o tempo de execução de uma instrução de máquina depende do estado do processador quando ela for executada
- Informações sobre o estado do processador é obtida através da análise dos potenciais caminhos de execução que levaram até aquela instrução
- Diferentes estados nos quais a instrução pode ser executada podem levar a tempos de execução muito variáveis

- Suponha que um laço itere 100 vezes
- O tempo de execução do corpo do laço no pior caso é 1ms
- Mas ele ocorre apenas uma vez, nas outras 99 vezes a memória cache foi povoada com os blocos de memória do laço e o tempo de execução no pior caso é de 0,5ms
- O tempo total no pior caso sem análise de cache seria estimado em 100ms
- Mas com análise de cache o tempo total no pior caso seria estimado em
  1 x 1ms + 99 x 0.5ms = 50,5ms

Análise do Comportamento do Processador 4/10

- A precisão da estimativa é melhorada se forem consideradas as possíveis histórias de execução que aconteceram até então
  - Definindo assim os contextos de execução possíveis para cada instrução

- A história de execução é o caminho que a execução seguiu até chegar em um ponto específico do programa
  - Laços, desvios, chamadas de subrotinas, etc

- Sempre que existir dúvida sobre o contexto de execução
  - Uma suposição pessimista deve ser empregada
  - Ou todas as possibilidades precisam ser exploradas na análise

Análise do Comportamento do Processador 5/10

- O contexto de execução inclui conhecimento estático sobre
  - O conteúdo da memória cache
  - A ocupação das unidades funcionais do pipeline
  - O estado interno do previsor de saltos (branch-predictor)
  - Etc

- Este conhecimento é usado para
  - Determinar se a posição de memória acessada está ou não na cache
  - Determinar se haverá algum atraso (stall) no pipeline
  - Determinar se o previsor de saltos irá acertar ou errar sua previsão
  - Etc

Análise do Comportamento do Processador 6/10

- A análise do comportamento do processador reune informações sobre o comportamento do processador para uma dada tarefa
  - Especificamente o comportamento dos componentes que influenciam os tempos de execução
- A análise do comportamento do processador requer portanto um modelo da arquitetura do processador
  - Não precisa incluir todos os detalhes da funcionalidade do processador
  - Basta um modelo simplificado que seja seguro com respeito aos tempos de execução
- A análise depende da corretude temporal do modelo usado

Análise do Comportamento do Processador 7/10

- Em geral os fabricantes de processadores não divulgam informações detalhadas sobre a organização interna do processador (microarquitetura)
- Quem faz a análise estática precisa desenvolver e validar o modelo do processador que será usado

- Esta validação pode ser feita com medições e observações
  - Tempos de execução medidos são comparados com os tempos de execução previstos
  - Eventos observados na execução real são comparados com eventos previstos
- A validação sempre será limitada
  - Ela é capaz de mostrar a presença de erros
  - Ela não é capaz de provar a ausência de erros

Análise do Comportamento do Processador 8/10

- A complexidade da análise do comportamento do processador obviamente depende da complexidade da arquitetura do processor

- Os microprocessadores mais poderosos sofrem o que é chamado de **anomalia temporal** (***timing anomalies***)
  - Anomalias são influências contra-intuitivas da execução de uma certa instrução de máquina no tempo de execução total da tarefa
  - São causadas pela presença de muitas características incluídas para melhorar o desempenho de forma probabilistica e que acabam afetando umas às outras

- Neste caso a construção de modelo temporal apropriado é muito difícil, beirando o impraticável

Análise do Comportamento do Processador 9/10

- Por exemplo, suponha que não é sabido se a próxima instrução está ou não na memória cache
  - Se a instrução X não estiver na cache haverá uma falta da cache (*cache-miss*) e o tempo de execução será maior
  - Se a instrução X estiver na cache (*cache-hit*) o acesso será rápido e o tempo de execução será menor
  - A intuição sugere que o segundo caso (*cache-hit*) sempre levará a um tempo menor de execução total para a tarefa

- Em processadores com **anomalias temporais** isto pode não acontecer
- A execução mais rápida (*cache-hit*) da instrução X pode levar a um tempo maior de execução para o total da tarefa em questão
  - Por exemplo, se a previsão de saltos (*branch-prediction*) errar
  - Um *cache-hit* aumentaria o prejuizo da previsão errada
  - Um *cache-miss* limitaria o prejuizo, gerando um tempo total menos

Análise do Comportamento do Processador 10/10

- **Anomalias temporais** violam suposições intuitivas
  - Que sempre supondo o pior caso local chegaremos no pior caso global
- Não é possível fazer a análise simplesmente buscando o pior caso para cada instrução de máquina individualmente
  - Isto não resulta no pior tempo de execução para a tarefa como um todo
- A existência de anomalias temporais em um processador limita os métodos de análise que podem ser aplicados
  - Torna a análise muito mais complexa

- A análise é forçada a seguir a execução por vários caminhos alternativos, pois o pior caminho não é conhecido a priori
  - Até ter certeza sobre qual o pior caminho
  - O espaço de busca torna-se muito grande

Análise da Memória Cache 1/2

- Impacto da memória cache é muito grande para ser ignorado
- Análise busca determinar qual é o conteúdo da memória cache a cada ponto da execução do programa
  - Fácil para um programa sem desvios e com o conteúdo inicial da cache conhecido, basta simular o comportamento da cache
  - Difícil no caso geral, quando o fluxo de controle depende dos dados de entrada
  - Pode-se chegar em um ponto do programa por vários caminhos diferentes, e cada um deles gera uma memória cache com conteúdo diferente
- Análise procura determinar aqueles dados ou instruções que com certeza estarão na memória cache (Always Hit) sempre que a execução passo por determinado ponto do programa
- No caso de dúvida, pessimismo

Análise da Memória Cache 2/2

- Análise da memória cache busca calcular o estado da cache para amenizar o pessimismo da análise
  - Super que sempre ocorre um cache-miss é muito pessimista

- Resultado da análise indica para cada bloco de memória qual a sua situação para cada ponto de execução do programa
  - Certamente não está na cache (Always Miss)
  - Certamente está na cache (Always Hit)
  - Dentro de um laço, não está na primeira iteração, porém estará nas demais iterações (First Miss)
  - Caminhos diferentes geram estados diferentes (Conflict), é necessário tratar como Always Miss por segurança (pessimismo)

Análise do Pipeline 1/5

- A análise do pipeline depende da complexidade do processador
- Em um pipeline simples a análise também será simples
- Suponha um processador onde:
  - O processador é livre de anomalias temporais
  - Não existe previsão de saltos (branch prediction)
  - Não existe execução fora de ordem
- Uma vez que inicia a execução do bloco básico, o comportamento do pipeline é contínuo e completamente previsível
- Podem haver atrasos no pipeline dentro do bloco básico
  - Quando uma instrução de máquina precisa do resultado da instrução de máquina anterior
  - Neste caso a execução da instrução de máquina que precisa do resultado vai ter que esperar
- Podem haver atrasos no pipeline entre blocos básicos
  - Quando a primeira instrução de máquina do bloco básico sucessor precisa do resultado da última instrução de máquina do bloco básico predecessor
  - Neste caso a execução da instrução de máquina que precisa do resultado vai ter que esperar

Análise do Pipeline 2/5

- A entrada em um bloco básico pode ocorrer de duas formas
- Acontece um salto para o início do bloco básico
  - Neste caso o bloco básico inicia com o pipeline vazio
  - Sua execução vai demorar mais
  - Basta somar o tempo de execução em separado dos dois blocos básicos
- A execução continua sequencialmente do final do bloco básico predecessor para o início do bloco básico sucessor
  - Neste caso o pipeline nunca parou de funcionar
  - As instruções iniciais do bloco básico sucessor começam a ser executadas nos estágios iniciais do pipeline enquanto as instruções finais do bloco básico predecessor são concluídas nos estágios finais do pipeline
  - Não é correto simplesmente somar os tempos de execução em separado dos dois blocos básicos

Análise do Pipeline 3/5

- Quando dois blocos básicos são executados com desvio, o tempo total de execução dos dois é a soma dos tempos em separado (7 + 6 = 13)

Clock	Busca	Decodifica	Busca operandos	Execução	Escrita
1	Bloco básico 1				
2	Bloco básico 1	Bloco básico 1			
3	Bloco básico 1	Bloco básico 1	Bloco básico 1		
4		Bloco básico 1	Bloco básico 1	Bloco básico 1	
5			Bloco básico 1	Bloco básico 1	Bloco básico 1
6				Bloco básico 1	Bloco básico 1
7					Bloco básico 1
8	Bloco básico 2				
9	Bloco básico 2	Bloco básico 2			
10		Bloco básico 2	Bloco básico 2		
11			Bloco básico 2	Bloco básico 2	
12				Bloco básico 2	Bloco básico 2
13					Bloco básico 2

Análise do Pipeline 4/5

- Quando dois blocos básicos são executados sem desvio, o tempo total de execução é menor que a soma dos tempos em separado (9 < 13)

Clock	Busca	Decodifica	Busca operandos	Execução	Escrita
1	Bloco básico 1				
2	Bloco básico 1	Bloco básico 1			
3	Bloco básico 1	Bloco básico 1	Bloco básico 1		
4	Bloco básico 2	Bloco básico 1	Bloco básico 1	Bloco básico 1	
5	Bloco básico 2	Bloco básico 2	Bloco básico 1	Bloco básico 1	Bloco básico 1
6		Bloco básico 2	Bloco básico 2	Bloco básico 1	Bloco básico 1
7			Bloco básico 2	Bloco básico 2	Bloco básico 1
8				Bloco básico 2	Bloco básico 2
9					Bloco básico 2
10					
11					
12					
13					

Análise do Pipeline 5/5

- A análise do pipeline precisa lidar com as características do pipeline no processador em questão
- O tempo de execução do bloco básico não depende apenas do pipeline
- É necessário considerar os acessos à memória cache que podem provocar atrasos no pipeline
- No caso de falta da cache é necessário acessar a memória principal muito mais lenta
  - E atualizar a memória cache

- Uma alteração em qualquer das características do processador requer uma revisão e possível adaptação da análise realizada
  - Adaptar o modelo abstrato do comportamento do processador

Cálculo do Limite Superior para o WCET 1/9

- O objetivo da análise estática é determinar um limite superior para o WCET da tarefa
- O limite superior precisa considerar o fluxo de controle e todas as informações obtidas nas outras etapas
- Existem três classes principais de métodos que podem ser usados para combinar analiticamente as informações obtidas
  - Baseados na estrutura (*structure-based*)
  - Baseados em caminhos (*path-based*)
  - Enumeração implícita de caminhos (*implicit-path enumeration*, IPET)

Cálculo do Limite Superior para o WCET 2/9

- **Nos Métodos Baseados na Estrutura**, um limite superior para o WCET é obtido combinando sucessivamente limites para o WCET de nodos próximos, conforme o tipo de comando que foi usado no programa original (IF, FOR, etc)
  - Nodos próximos do Grafo de Fluxo de Controle são agrupados em nodos únicos e o WCET estimado para o nodo agrupado resultante deve ser o comportamento de pior caso para o grupo de nodos agrupados
  - Para cada nodo precisa ser considerado o seu pior contexto de execução, ou seja, a história de execução que leva ao pior caso
  - Considerar diferentes contextos de execução pode requerer transformações no Grafo de Fluxo de Controle para refletir os diferentes contextos
  - O programa precisa ser programado de forma bem estruturada
  - A abordagem assume uma correspondência direta entre as estruturas do código fonte e o programa em código de máquina, não lidando bem com otimizações feitas pelo compilador

Cálculo do Limite Superior para o WCET 3/9

- **Nos Métodos Baseados em Caminhos**, um limite superior para o WCET da tarefa é determinado através do cálculo de limites para o WCET em cada caminho da tarefa, e da identificação do caminho que leva ao maior tempo de execução
  - Todos os caminhos de execução possíveis precisam ser representados explicitamente
  - O método baseado em caminho é viável na ausência de laços
  - Porém encontra problemas quando existem laços aninhados
  - O número de caminhos cresce exponencialmente com o número de pontos de desvio no programa

Cálculo do Limite Superior para o WCET 4/9

- O método mais usado é a **Enumeração Implícita de Caminhos** (*Implicit-Path Enumeration*, IPET)
  - O fluxo do programa e os limites para o WCET da execução de cada bloco básico são combinados em conjuntos de restrições aritméticas
  - Para cada bloco básico e fluxo de execução (entidade) é definido:
  - Um coeficiente de tempo (**time coefficient**), que expressa um limite superior para a contribuição daquela entidade para o tempo total de execução, toda vez que ela é executada
  - Uma variável contadora (**count variable)**, que corresponde ao número de vezes que a entidade é executada

Cálculo do Limite Superior para o WCET 5/9

- Um limite superior para o WCET da tarefa é obtido através da maximização da soma dos produtos de cada coeficiente de tempo (ti) pela sua respectiva variável contadora (xi)

- $\sum_{i \in entities} (x_i \, . \, t_i)$

- As variáveis contadoras estão sujeitas a restrições que refletem a estrutura da tarefa e os possíveis caminhos de execução
- Uma solução para o problema colocado pelo IPET é um limite superior para o tempo de execução da tarefa através da definição de valores para ti e xi que maximem o tempo de execução
- Para isto é usada a Programação Linear Inteira

Cálculo do Limite Superior para o WCET 6/9

- A **Programação Linear** é um método genérico para codificar os requisitos de um sistema na forma de um sistema de restrições lineares
  - Uma função objetivo deve ser maximizada ou minimizada para obter-se uma atribuição ótima de valores às variáveis do sistema
- Temos **Programação Linear Inteira** quando os valores envolvidos devem ser valores inteiros

- Um problema descrito através da Programação Linear Inteira é NP-hard
  - Ele tem uma complexidade computacional potencialmente exponencial com respeito ao tamanho da tarefa

- Logo, o uso de Programação Linear Inteira deveria ser limitado a pequenas instâncias de problemas ou subproblemas da análise estática
- Entretanto, ferramentas atuais conseguem resolver problemas de Programação Linear Inteira de tamanho considerado

Cálculo do Limite Superior para o WCET 7/9

- O fluxo de controle das tarefas é descrito como um problema de Programação Linear Inteira
- Informações adicionais sobre o controle de fluxo podem ser codificadas como restrições adicionais
- A função objetivo expressa o tempo de execução da tarefa completa
  - Este valor maximizado representa um limite superior para todos os tempos de execução possíveis para a tarefa

- Exemplos de ferramentas (solvers) para resolver problemas de Programação Linear Inteira são
  - GLPK, GNU Project - Free Software Foundation (FSF)
    - https://www.gnu.org/software/glpk/
  - CPLEX Optimizer, IBM
    - https://www.ibm.com/products/ilog-cplex-optimization-studio

Cálculo do Limite Superior para o WCET 8/9

- No IPET as características do fluxo de controle são traduzidas em restrições da Programação Linear Inteira
- As restrições estruturais refletem os possíveis fluxos de execução
- As restrições *start* e *exit* determinam que a tarefa deve ser iniciada exatamente uma vez e terminada exatamente uma vez
- Todo bloco básico é iniciado exatamente o mesmo número de vezes que ele é terminado (fluxo é conservado)
- Todo laço tem um limite para o número de iterações, o qual aparece como uma restrição no número de vezes que o bloco básico que representa o início do laço pode ser executado (laços não podem executar além deste limite)
- O bloco básico que avalia a condição de teste do laço deve executar uma vez a mais do que o corpo do laço

Cálculo do Limite Superior para o WCET 9/9

- Questões ligadas ao comportamento do processador (pipeline, cache, etc) aparecem nos valores de ti ou em restrições adicionais
  - Cada entidade pode ser simplesmente um bloco básico
  - Ou cada entidade pode ser um bloco básico acessado a partir de outro bloco básico específico
  - Ou cada entidade pode ser um bloco básico acessado a partir de um caminho específico
- A modelagem vai depender da complexidade do processador em questão, precisa incluir compensações para os efeitos do pipeline

- Objetivo da Programação Linear Inteira é maximizar
- $\sum_{i \in entities} (x_i \, . \, t_i)$
- Definindo valores para xi que respeitem as restrições colocadas

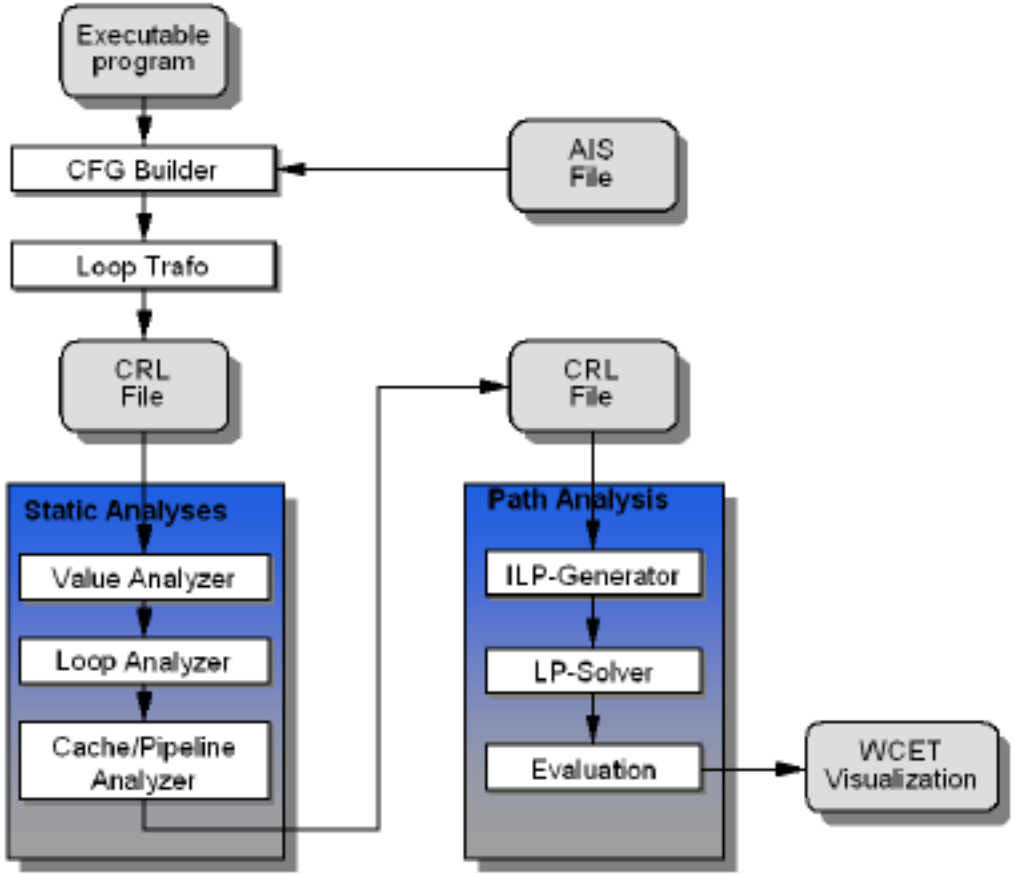
A Ferramenta aiT 1/8

- Desenvolvido pela empresa AbsInt, aiT é o analisador estático de WCET comercial mais usado pela indústria
  - www.absint.com/ait

- O aiT analisa diretamente código de máquina e considera o comportamento de elementos tais como cache e pipeline
- Um conjunto de ferramentas é fornecido, inclusive com interface gráfica de usuário para diversos aspectos

A Ferramenta aiT 2/8

- O aiT, como toda análise estática de WCET, emprega várias fases no processo (www.absint.com/ait)



A Ferramenta aiT 3/8

- A construção do GFC (CFG Buider) identifica as instruções de máquina do processador em questão e reconstrói o Grafo de Fluxo de Controle a partir do código binário
- A decodificação das instruções de máquina é capaz de identificar o endereço alvo de saltos que utilizem endereços absolutos ou relativos ao program counter
  - Tem dificuldades com endereços alvos computados a partir do conteúdo de registradores
  - São usados decodificadores especializados que sabem como alguns compiladores geram o código binário
  - Pode ser necessário que o programador informe através de uma anotação qual o endereço alvo do salto

A Ferramenta aiT 4/8

- A análise de valor (Value Analysis) computa intervalo de valores para registradores e intervalos de endereços para as instruções que acessam a memória
- A análise de valor tenta determinar os valores nos registradores para cada ponto do programa em cada caminho possível
  - um limite inferior e um limite superior
- São usados para determinar os endereços alvo no caso de acesso indireto à memória (via ponteiros)
  - Para a análise da cache
- São usados para determinar limites para o número de iterações de laços
- O que não poder ser determinado pela análise de valor deverá ser fornecido pelo usuário na forma de anotações

A Ferramenta aiT 5/8

- A análise de laços (Loop Bound Analysis) determina limites superior para o número de iterações de laços simples
- A análise de laços utiliza a análise de valor e também o reconhecimento de padrões de código gerados pelos compiladores
- Anotações do usuário devem fornecer os limites que não puderem ser determinados automaticamente.

A Ferramenta aiT 6/8

- A análise da cache (Cache Analysis) classifica as referências à memória como acerto (cache-hit) ou falta (cache-miss)

- A análise considera caches que utilizam a política de substituição LRU (Least Recently Used)

- Também outras políticas de substituição tais como pseudo-roundrobin (ColdFire MCF 5307) e pseudo-LRU (PLRU, PowerPC MPC 750 e 755)

A Ferramenta aiT 7/8

- A análise do pipeline (Pipeline Analysis) prevê o comportamento do programa no pipeline do processador
- O resultado é o tempo de execução para cada bloco básico
- Para as as várias histórias de execução possíveis até chegar no bloco básico em questão
- Os resultados da análise da cache são usados pela análise do pipeline
  - Permite identificar atrasos no pipeline (pipeline stalls) devidos às faltas da cache

A Ferramenta aiT 8/8

- A análise de caminhos (Path Analysis) determina o caminho que leva ao tempo de execução no pior caso
- A análise de caminhos emprega o método IPET descrito antes
- A estimativa do WCET da tarefa é obtida através da Programação Linear Inteira
- As análises de valor, de cache e de pipeline utilizam Interpretação Abstrata
  - Um método baseado em semântica para a análise estática de programas

Considerações Finais 1/2

- A análise estática calcula um limite superior para o tempo de execução no pior caso
- A análise de fluxo de controle cobre todos os caminhos de execução possíveis
- Também são cobertos todas as possíveis dependências de contexto de execução do comportamentodo processador

- O aspecto negativo é a necessidade de modelos específicos para o comportamento do processador
- Os resultados são apenas aproximados, no caso um limite superior pessimista para o WCET

- Um aspecto positivo é que a análise pode ser feita sem precisar realmente executar a tarefa analisada

Considerações Finais 2/2

- As técnicas para estimação do WCET via análise estática acabam sendo limitadas no que se refere a
  - plataformas de hardware suportadas
  - estilo de programação tolerado
- A maior dificuldade técnica da análise estática é a modelagem do comportamento do processador
  - O modelo é sempre específico para um processador e uma plataforma de hardware

- Este é um problema que não existe na maioria dos métodos baseados em medição
- Implementar um método de medição para um novo processador é muito mais fácil do que criar um modelo abstrato do seu comportamento
- Entretanto, uma medição é apenas uma aproximação otimista do WCET
- A não ser que algum tratamento estatístico seja aplicado

Sumário

- Introdução
- Importância do Tipo de Hardware
- Análise Estática do WCET
- Análise do Fluxo de Controle
- Análise do Comportamento do Processador
- Análise da Memória Cache
- Análise do Pipeline
- Cálculo do Limite Superior para o WCET
- A Ferramenta aiT
- Considerações Finais