Trabalho Arquitetura III:

Integrantes:

- Gustavo Fonseca
- Gustavo Luiz
- Ricado Sena

Belo Horizonte, 21 de Novembro de 2016

Tema:

Análise da eficiência de predições de desvios em cargas de trabalho destintas.

-O que é predição de dados
-Arquitetura Proposta
-Carga de Trabalho
-Resultados e Conclusões
-Trabalhos Futuros

Relembrando...

A predição de desvio é uma técnica utilizada no pipeline escalar e super escalar, para evitar que instruções que poderiam ser executadas no nível paralelo de instrução fiquem ociosas (abandonadas), ou seja, podemos aproveitar etapas da execução ainda mais usando a predição no pipeline.

Relembrando...

A predição de desvio é uma técnica utilizada no pipeline escalar e super escalar, para evitar que instruções que poderiam ser executadas no nível paralelo de instrução fiquem ociosas (abandonadas), ou seja, podemos aproveitar etapas da execução ainda mais usando a predição no pipeline.

Com isso teremos:

Relembrando...

A predição de desvio é uma técnica utilizada no pipeline escalar e super escalar, para evitar que instruções que poderiam ser executadas no nível paralelo de instrução fiquem ociosas (abandonadas), ou seja, podemos aproveitar etapas da execução ainda mais usando a predição no pipeline.

Com isso teremos:

Menor frequência de conflito de controle.

Relembrando...

A predição de desvio é uma técnica utilizada no pipeline escalar e super escalar, para evitar que instruções que poderiam ser executadas no nível paralelo de instrução fiquem ociosas (abandonadas), ou seja, podemos aproveitar etapas da execução ainda mais usando a predição no pipeline.

Com isso teremos:

Menor frequência de conflito de controle.

Número maior de IPC sem modificar o tamanho da memória

Relembrando...

A predição de desvio é uma técnica utilizada no pipeline escalar e super escalar, para evitar que instruções que poderiam ser executadas no nível paralelo de instrução fiquem ociosas (abandonadas), ou seja, podemos aproveitar etapas da execução ainda mais usando a predição no pipeline.

Com isso teremos:

Menor frequência de conflito de controle.

Número maior de IPC sem modificar o tamanho da memória

Maior IPC implica em
Maior Desempenho

Arquitetura proposta

Para a realização de testes foi utilizado um processador SMP com 8 núcleos sendo que:

- Memória L2 com 512kB associativa 8-way
- Memória L1 com 128kB associativa 4-way
- Política de escrita: *Write Through*
- Política de substuição tanto em L2 e em L1 foi selecionado o LRU (*Least Recently Used*).

Simulação

A simulação foi realizada no SESC por ser uma plataforma que possui uma melhor descrição dos componentes a serem modficiados na arquitetura.

Benchmark

- O *Ocean* foi o *benchmark* escolhido para a realização dos testes.
- O *Ocean* executa uma série de operações em uma matriz nxn de elementos que simulam um oceano.

Benchmark

- O *Ocean* foi o *benchmark* escolhido para a realização dos testes.
- O *Ocean* executa uma série de operações em uma matriz nxn de elementos que simulam um oceano.

Quanto maior o n, mais “pesado” o programa fica.

Cargas de trabalho

Ocean

- 4x4
- 10x10
- 18x18
- 34x34
- 66x66

$2^n + 2$ Com n > 1 e n inteiro

E n > 2 para multi cores.

Cargas de trabalho

Ocean

- 4x4
- 10x10
- 18x18
- 34x34
- 66x66

Com a variedade das cargas de trabalho, é interessante pra verificar a eficiência em situações diferentes.

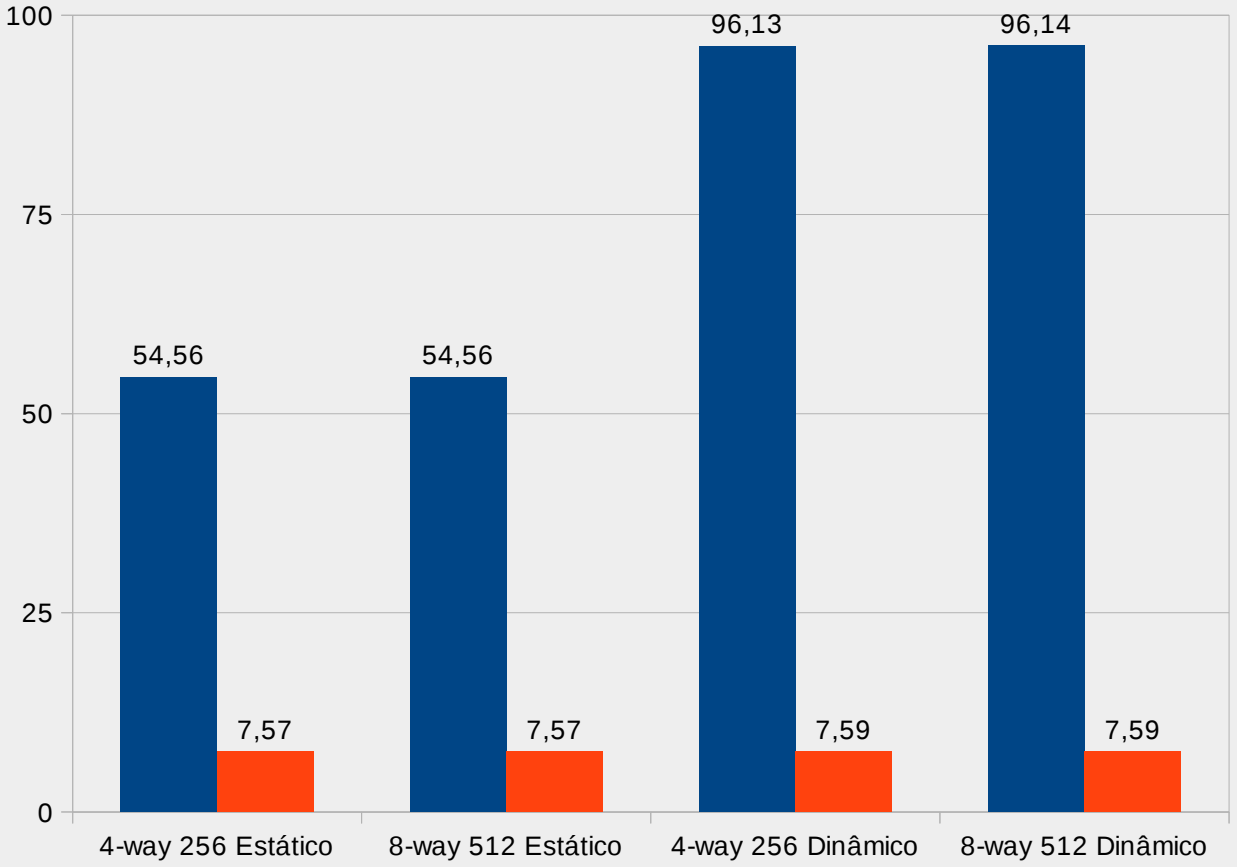
$2^n + 2$ Com n > 1 e n inteiro

E n > 2 para multi cores.

Tipos de predição que serão analisados

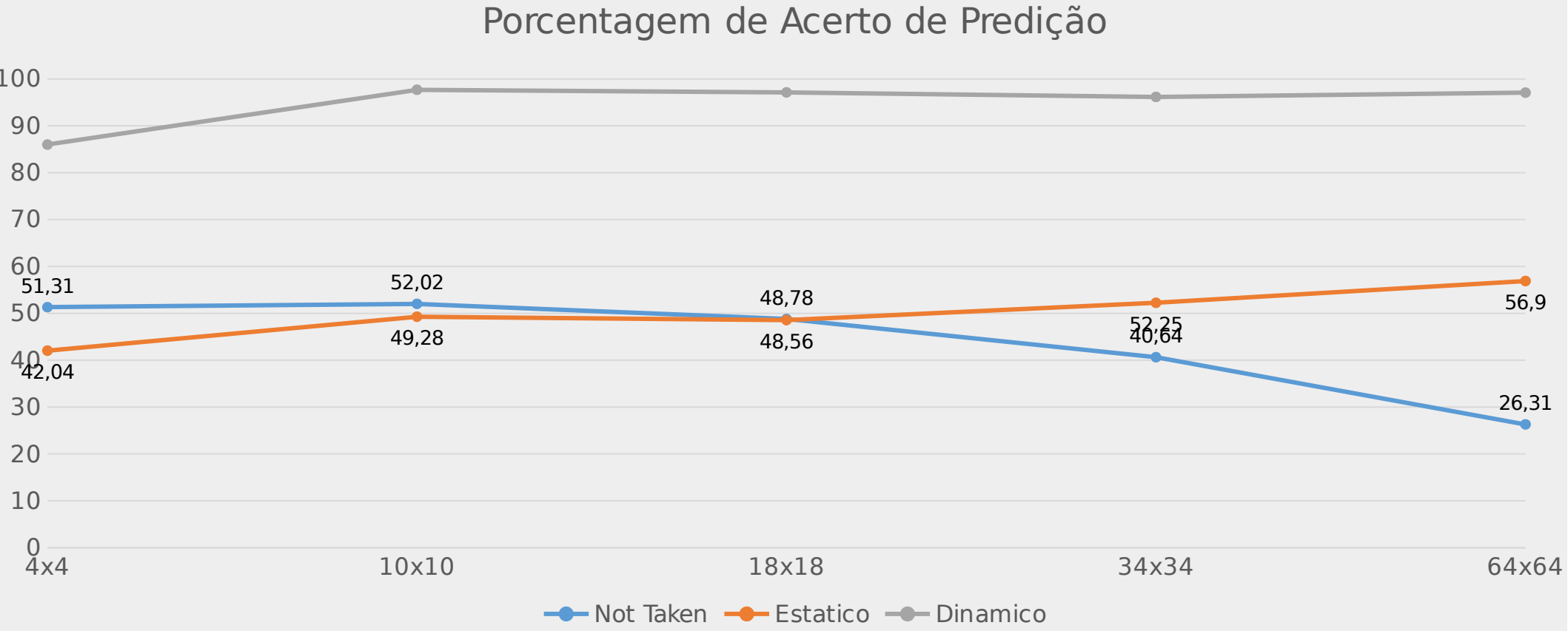
- Estático (*Static*) : Aonde assume-se que haverá um desvio.
- Estático (*NotTaken*): Variação do *Static* que assume que **não** haverão desvios.
- Dinâmico(Hybrid) : Aonde o compilador poderá escolher entre uma tabela ou um histórico.

Relação da associatividade com a predição de desvio.



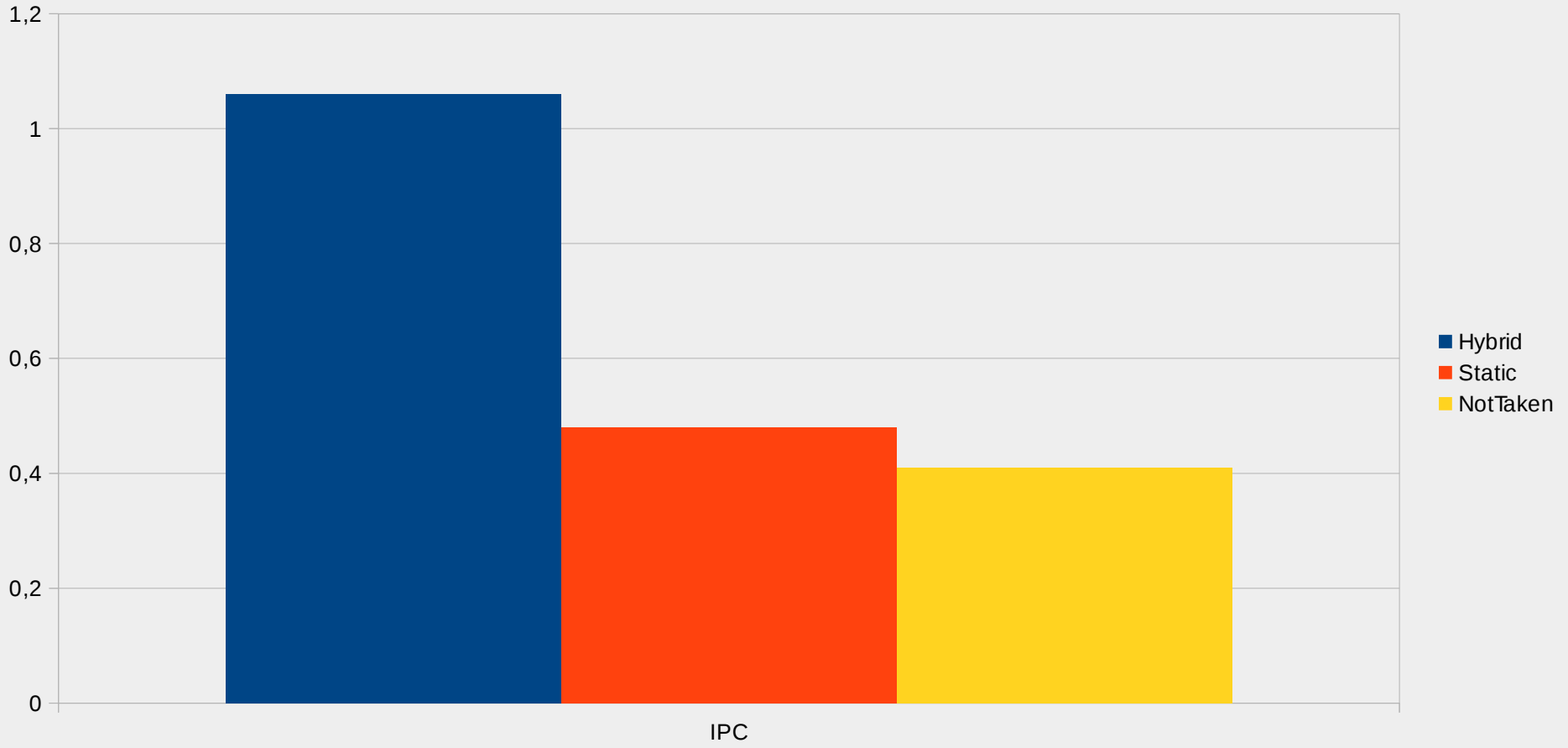
Podemos perceber pelo gráfico que não adianta aumentar o tamanho da cache e/ou associatividade.

A Taxa será praticamente a mesma!!

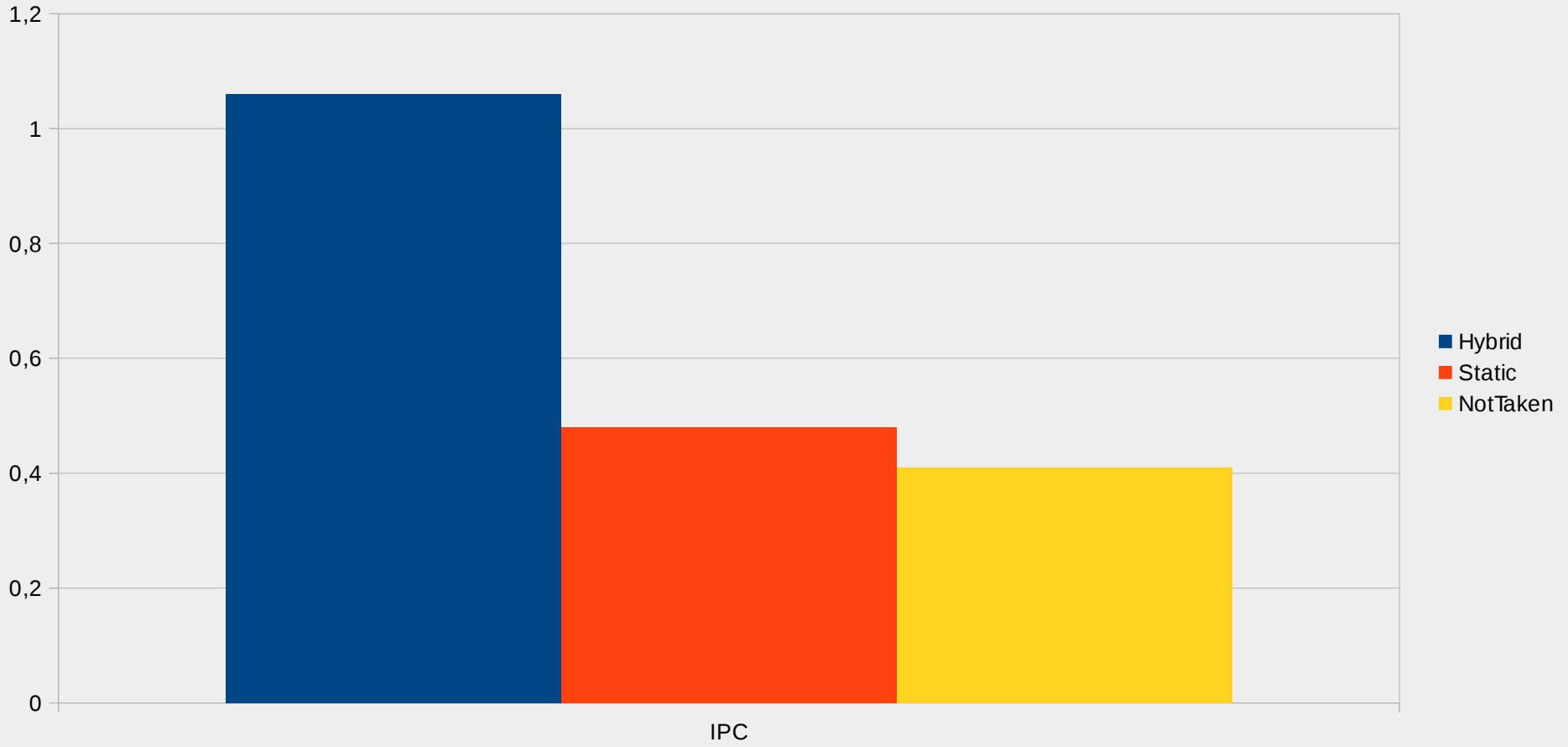


No *ocean* 4x4 foi utilizado apenas o processamento de 1 núcleo pois por possuir um tamanho pequeno não possui o suporte para o processamento de um número maior de núcleos.

Taxa IPC



Taxa IPC



Podemos ver claramente que a taxa do IPC aumentou relativo à porcentagem de acertos no Branch

Tempo de execução

66x66	Exe Speed	Exe MHz	Exe Time	Sim Time (5000MHz)
Hybrid	751.137 KIPS	0.0919 MHz	57.820 secs	1.062 msec
Static	626.135 KIPS	0.1652 MHz	62.210 secs	2.055 msec

Em casos aonde o benchmark é mais "pesado" o *Hybrid* foi melhor que o *Static*, mas em um benchmark leve...

Tempo de execução

66x66	Exe Speed	Exe MHz	Exe Time	Sim Time (5000MHz)
Hybrid	751.137 KIPS	0.0919 MHz	57.820 secs	1.062 msec
Static	626.135 KIPS	0.1652 MHz	62.210 secs	2.055 msec
10x10				
Not-Taken	675.453 KIPS	0.3227 MHz	4.620 secs	0.298 msec
Hybrid	812.320 KIPS	0.1876 MHz	6.440 secs	0.242 msec

Podemos perceber que em arquiteturas menores aonde não ocorrem muitas comparações, mesmo com o IPC menor, o *Not-Taken* consegue um tempo de execução menor.

Conclusão

Conseguimos perceber que a melhora no IPC é possível ao utilizar a predição de desvios. Mas é importante analisar para cada arquitetura e carga de trabalho qual utilizar, mas é evidente que a Predição dinâmica híbrida obteve um resultado muito melhor do que o Static e NotTaken em situações de carga maior (como 64x64 e 34x34).

Trabalhos futuros

- Análise feita em outras arquiteturas (Redes em chip)
- Realizar testes com mais tipos de predições e trabalhar com outras situações (politica de cache, threads, etc.)

Dúvidas??

Obrigado

Live for nothing or die for something. Rambo IV(2008)

Artigo

Análise da eficiência de predições de desvios em cargas de trabalho distintas

Gustavo Luiz Tavares
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Brasil

Gustavo Henrique Moreira Fonseca
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Brasil

Ricardo Xavier Sena
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Brasil

***Abstract*—This article is a study of different types of Branch Predction and it's impact on perfomance. We used three types of branch predction to demonstrate how each feature will impact on instructions per cicle. We also used SESC (Super ESCalar Simulator) to simulate the CPU architecture and test it's efficiency using Ocean Benchmark with five sizes.**

Keywords—Branch; Pipeline; SESC

I. INTRODUÇÃO

A predição de desvio é uma técnica utilizada nos pipelinas para que as intruções que poderiam ser executadas a nível paralelo não fiquem ociosos, esperando a instrução anterior ser executada. Esse método pode impactar em um maior desempenho na execução, pois implica em um número maior de instruções executadas em paralelo à nivel de instrução, sem realizar maiores modificações na parte de *hardware.* Este artigo realiza uma análise da perfomance em dois tipos de predições de desvio utilizando cargas de trabalho variados, para verificar o comportamento de cada situação possível.

II. METODOLGIA

A. Ambiente de trabalho e Benchmark

Fpara a realização dos testes, foi utilizado como simulador o SESC para a utilização do esboço de arquitetura e também para o benchmark por alguns motivos dentre eles:

- Facilidade de configuração da arquitetura;
- Quantidade razoável de documentações
- Bom suporte ao UNIX
- Simula o processador de forma bastante variada, permitindo assim o melhor aprofundamento.

Acerca do benchmark foi utilizado o Ocean que realiza uma análise de movimentos no oceano baseado no tamanho informado por matriz. Na análise serão usados 5 tamanhos diferentes: 4x4, 10x10, 18x18, 34x34 e 66x66.

B. Mespecificações do processador

Foi utilizado para testes um processador com arquitetura SMP (*Symmetric Multi-Processing)* com 8 núcleos e sua cache distribuída em:

- L1: 128kB separados em 64kB para dados e os outros 64kB para instrução, e mapeamento do tipo associativa 4-way
- L2: 512kB privados com mapeamento do tipo associativa 8-way.

Para a política foi utilizado o tipo de escrita Write Through e a política de substuição LRU

C. *Avaliação do desempenho*

Para a realização dos testes de predição de desvios, serão utilizados três tipos, sendo deles dois Estáticos (*Static, NotTaken*) e apenas um dinâmico (*Hybrid).* A utilização de dois formatos de predições do mesmo tipo é pelo fato de que a característica que os definem podem impactar de forma diferente no processamento.

III. Predições de desvio

Como dito anteriormente, os dois tipos de predição de desvio que será feito a análise serão a estática e dinâmica. Eles possuem algumas diferenças entre eles.

A. Predição Estática.

A predição estática funciona de uma maneira bem simples, o processador ele irá assumir se o desvio será realizado ou não. Neste caso para as duas variações será feito da seguinte forma :

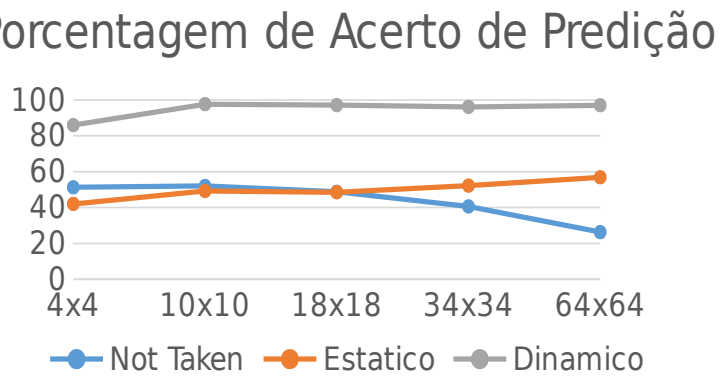
- Static : O desvio sempre será tomado
- NotTaken : O desvio não será tomado

B. Predição Dinâmica

A predição dinâmica possui uma forma de utilização um pouco mais complexa. O processador ele ao perceber um desvio verifica em uma tabela criada com as instruções que já ocorreram desvio (dicionário) e caso for possível realizar o desvio tendo como parâmetro o bit de validade é assumido atomada do desvio. Caso não exista a instrução verificada na tabela, ela é inserida e não ocorre o desvio.

IV. Testes e resultados

Após os testes realizados, foi contestado que a predição dinâmica por possuir uma lógica mais robusta, possuiu um desempenho muito maior do que a predição estática como mostra o gráfico abaixo:



Fonte: Elaborado pelo Autor

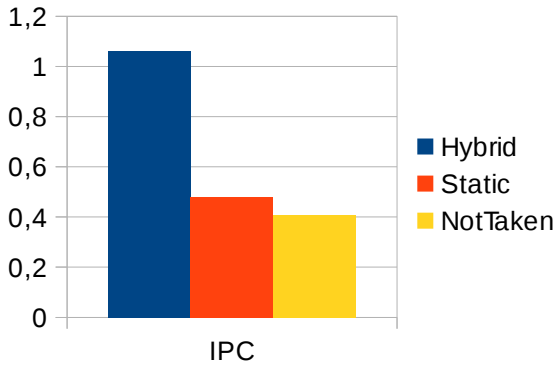
A predição dinâmica possuiu o acerto de 97,06 % na matriz de oceano de maior valor (64x64). Já o desempenho do estático ficou bem abaixo, sendo 56,90% e 26,31% para o *Static* e o *Not-Taken* respectivamente. Vale ressaltar que os testes realizados na matriz 4x4 por ser muito pequena foi utilizado apenas um dos cores, devido ao baixo tamanho.

A capacidade de melhora em benchmarks maiores na utilização da predição Dinâmica se deve ao fato de que com o alto número de instruções, a tendência é de que a quantidade de desvios e instruções repetidos aumenta, sendo assim a facilidade de tomada de decisão na parte final do benchmark, enquanto o *Not-Taken* precisa fazer a análise, mesmo as que se repetem, dos desvios que não irão assumir o desvio.

Conseguimos perceber também que devido ao maior número de processos que são executados, uma arquitetura utilizando a predição dinâmica irá possuir um IPC relativamente maior do que a predição estática, sendo de 1,07 Instruções por siiclo enquanto o *Static* possuiu um um IPC de 0,48, e o *Not-Taken* 0,41.

Também percebe-se que, mesmo com a diferença entre as predições estáticas quanto a porcentagem de acerto, não refletiu de forma positiva no aumento da taxa de IPC, sendo apenas de 0,07. Como mostra o gráfico abaixo:

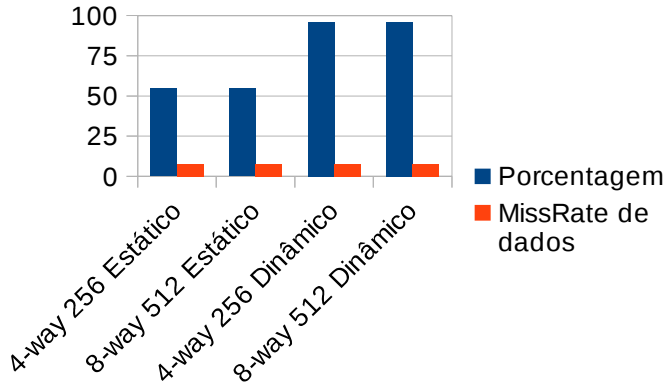
Gráfico II: Amostragem de IPC baseado nos testes feito com matriz 64x64



Fonte: Elaborado pelo autor

O fato de possuir um IPC bem superior, faz com que a Predição dinâmica seja bem mais utilizada, mas podemos perceber que por ter um gasto de memória maior, pode ocorrer Cache Miss relativo à conflitos em memória em casos aonde a memória oferecida é inferior. Utilizando uma memória cache L1 com 32kB(ou menos) e com uma associatividade n-way, com n > 8. Percebe-se que há uma taxa elevada de CacheMiss na parte dos dados superior à 15%, enquanto na arquitetura proposta a média não oscilou mesmo com o aumento do tamanho da memória e associatividade, como mostra o gráfico a seguir:

Gráfico III: Amostragem de Acerto e Miss rate de dados baseado nos testes feitos utilizando ocean com matriz 64x64



Conclusão

Podemos concluir com que pela análise realizada confirma a melhoria na quantidade de do paralelismo na arquitetura apenas por mudar o tipo de predição de desvio do tipo dinâmico. Entretanto em casos aonde não há o suporte da predição dinâmica, entre os casos estudados da predição estática, irá depender do engenheiro de software escolher para que tipo de aplicação será utilizado, já que existe uma diferença de eficiência baseada no tamanho da carga de trabalho..

Referências

http://www.manio.org/blog/sesc-tutorial-1-instaall-whole-sesc-step-by-step/

http://www.netsoft.inf.br/aulas/1_SIN_Arquitetura_de_Computadores/Predicao.pdf

```
#Numero de nucleos
procsPerNode  = 8
cacheLineSize = 32

issue         = 4     # processor issue width
cpucore[0:$(procsPerNode)-1] = 'issueX'

#<shared.conf> (contents below)

###############################
# SYSTEM                      #
###############################

enableICache   = true
NoMigration    = true
tech           = 0.10
pageSize       = 4096
fetchPolicy    = 'outorder'
issueWrongPath = true

technology = 'techParam'

###############################
# clock-panalyzer input       #
###############################
[techParam]
clockTreeStyle = 1              # 1 for Htree or 2 for balHtree
tech       = 70             # nm
frequency  = 5e9                # Hz
skewBudget = 20                         # in ps
areaOfChip = 200                # in mm^2
loadInClockNode = 20                    # in pF
optimalNumberOfBuffer = 3

###############################
# PROCESSORS' CONFIGURATION  #
###############################

[issueX]
frequency       = 5e9
areaFactor      = ($(issue)*$(issue)+0.1)/16  # Area compared to Alpha264 EV6
inorder         = false
fetchWidth      = $(issue)
instQueueSize   = 2*$(issue)
issueWidth      = $(issue)
retireWidth     = $(issue)+1
decodeDelay     = 6
renameDelay     = 3
wakeupDelay     = 6               # -> 6+3+6+1+1=17 branch mispred. penalty
maxBranches     = 16*$(issue)
bb4Cycle        = 1
maxIRequests    = 4
```

```
interClusterLat = 2
intraClusterLat = 1
cluster[0]      = 'FXClusterIssueX'
cluster[1]      = 'FPClusterIssueX'
stForwardDelay  = 2
maxLoads        = 10*$(issue)+16
maxStores       = 10*$(issue)+16
regFileDelay    = 3
robSize         = 36*$(issue)+32
intRegs         = 32+16*$(issue)
fpRegs          = 32+12*$(issue)
bpred           = 'BPredIssueX'
dtlb            = 'FXDTLB'
itlb            = 'FXITLB'
dataSource      = "DMemory DL1"
instrSource     = "IMemory IL1"
enableICache    = true
OSType          = 'dummy'


# integer functional units

[FXClusterIssueX]
winSize    = 12*$(Issue)+32 # number of entries in window
recycleAt  = 'Execute'
schedNumPorts = 4
schedPortOccp = 1
wakeUpNumPorts= 4
wakeUpPortOccp= 1
wakeupDelay   = 3
schedDelay    = 1 # Minimum latency like a intraClusterLat
iStoreLat  = 1
iStoreUnit = 'LDSTIssueX'
iLoadLat   = 1
iLoadUnit  = 'LDSTIssueX'
iALULat    = 1
iALUUnit   = 'ALUIssueX'
iBJLat     = 1
iBJUnit    = 'ALUIssueX'
iDivLat    = 12
iDivUnit   = 'ALUIssueX'
iMultLat   = 4
iMultUnit  = 'ALUIssueX'

[LDSTIssueX]
Num = $(issue)/3+1
Occ = 1

[ALUIssueX]
Num = $(issue)/3+1
Occ = 1
```

```
# floating point functional units

[FPClusterIssueX]
winSize    = 8*$(issue)
recycleAt  = 'Execute'
schedNumPorts = 4
schedPortOccp = 1
wakeUpNumPorts= 4
wakeUpPortOccp= 1
wakeupDelay   = 3
schedDelay    = 1 # Minimum latency like a intraClusterLat
fpALULat   = 1
fpALUUnit  = 'FPIssueX'
fpMultLat  = 2
fpMultUnit = 'FPIssueX'
fpDivLat   = 10
fpDivUnit  = 'FPIssueX'

[FPIssueX]
Num = $(issue)/2+1
Occ = 1

# branch prediction mechanism

#Tipos de predicao

#BPeedIssue type
#Suportados : Static, Not-Taken, 2Level,2bit,Taken,Oracle,2BcgSkew,YAGS,Ogehl,Hybrid
#Utilizados : Static, Not-Taken, Hybrid
[BPredIssueX]
type          = "Hybrid"
bits=2
BTACDelay     = 0
l1size        = 1
l2size        = 16*1024
l2Bits        = 1
historySize   = 11
Metasize      = 16*1024
MetaBits      = 2
localSize     = 16*1024
localBits     = 2
btbSize       = 2048
btbBsize      = 1
btbAssoc      = 2
btbReplPolicy = 'LRU'
btbHistory    = 0
rasSize       = 32

# memory translation mechanism

[FXDTLB]
deviceType = 'cache'
```

```
size       = 64*8
assoc      = 4
bsize      = 8
numPorts   = 2
replPolicy = 'LRU'

[FXITLB]
deviceType = 'cache'
size       = 64*8
assoc      = 4
bsize      = 8
numPorts   = 2
replPolicy = 'LRU'

################################
# MEMORY SUBSYSTEM           #
################################

# instruction source
[IMemory]
deviceType    = 'icache'
size          = 32*1024
assoc         = 2
bsize         = $(cacheLineSize)
writePolicy   = 'WT'
replPolicy    = 'LRU'
numPorts      = 2
portOccp      = 1
hitDelay      = 2
missDelay     = 1               # this number is added to the hitDelay
MSHR          = "iMSHR"
lowerLevel    = "L1L2Bus L1L2"

[iMSHR]
type = 'single'
size = 32
bsize = $(cacheLineSize)

# data source
[DMemory]
deviceType    = 'cache'
size          = 32*1024
assoc         = 4
bsize         = $(cacheLineSize)
writePolicy   = 'WT'
replPolicy    = 'LRU'
numPorts      = $(issue)/3+1
portOccp      = 1
hitDelay      = 2
missDelay     = 1               #this number is added to the hitDelay
maxWrites     = 8
MSHR          = "DMSHR"
```

```
lowerLevel    = "L1L2Bus L1L2"

[DMSHR]
type = 'single'
size = 64
bsize = $(cacheLineSize)

# bus between L1s and L2
[L1L2Bus]
deviceType = 'bus'
numPorts   = 1
portOccp   = 1                  # assuming 256 bit bus
delay      = 1
lowerLevel = "L2Cache L2 sharedBy 1"



#tipos de cache L2 utilizada
#8way LRU, 8way RANDOM, 16way LRU, 16way RANDOM

#replPolicy    = 'LRU' Tipo substituicao associativa
#assoc         = 8

# private L2
[L2Cache]
deviceType    = 'smpcache'
size          = 512*1024
assoc         = 8
bsize         = $(cacheLineSize)
writePolicy   = 'WB'
replPolicy    = 'LRU'
protocol      = 'MESI'
numPorts      = 2               # one for L1, one for snooping
portOccp      = 2
hitDelay      = 9
missDelay     = 11               # exclusive, i.e., not added to hitDelay
displNotify   = false
MSHR          = 'L2MSHR'
lowerLevel    = "SystemBus SysBus sharedBy 16"

[L2MSHR]
size =  64
type = 'single'
bsize = $(cacheLineSize)

[SystemBus]
deviceType    = 'systembus'
numPorts      = 1
portOccp      = 1
delay         = 1
lowerLevel    = "MemoryBus MemoryBus"
BusEnergy     = 0.03
```

```
[MemoryBus]
deviceType    = 'bus'
numPorts      = 1
portOccp      = $(cacheLineSize) / 4   # assuming 4 bytes/cycle bw
delay         = 15
lowerLevel    = "Memory Memory"

[Memory]
deviceType    = 'niceCache'
size          = 64
assoc         = 1
bsize         = 64
writePolicy   = 'WB'
replPolicy    = 'LRU'
numPorts      = 1
portOccp      = 1
hitDelay      = 500 - 31  # 5.0GHz: 100ns is 500 cycles RTT - 16 busData
missDelay     = 500 - 31  # - 15 memory bus => 500 - 31
MSHR          = NoMSHR
lowerLevel    = 'voidDevice'

[NoMSHR]
type = 'none'
size = 128
bsize = 64

[voidDevice]
deviceType    = 'void'

############################
#     BEGIN MIPSEMUL       #
############################

[FileSys]
mount=""
```