Fundamentos de Arquitetura de Computadores

Tiago Alves

Faculdade UnB Gama
Universidade de Brasília

Benchmarks

Avaliação de desempenho ideal: capaz de discriminar/prever diferenças de desempenho na aplicação real

Workload: Conjunto de programas típicos que caracterizam a utilização da máquina

- Depende do usuário (ex. engenharia, desenvolvimento, finanças, jogos, etc)
- Difícil padronização para fins de comparação

Benchmarks: Conjuntos de programas especificamente escolhidos para medir o desempenho em determinada categoria de aplicações.

Workload que o usuário espera que represente mais aproximadamente o desempenho no workload real!

Benchmarks

Real:

- Aplicações reais
- Grande complexidade e variação das instruções
- Difícil de medir pois depende muito do fluxo de I/O

Sintético:

- Pequenos fragmentos de código.
- Fáceis de padronizar, aplicar e medir
- Adequado na etapa de desenvolvimento (arquitetura, compilador)
- Facilmente otimizável (forçado)

SPEC

System Performance Evaluation Cooperative

Tabela de testes para benchmarking. UUT 2.66 GHz Intel Core i7 920.

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Qual benchmark usar?

Tempo de resposta:

- Aplicações científicas, matemáticas, gráficas, etc
- Tempo de CPU

Vazão:

- Aplicações em servidores (Banco de dados, Web server, etc)
- *Throughput*

Exemplo Individual x Coletivo

Analise a tabela comparativa abaixo do tempo de execução de dois programas em três computadores diferentes.

	Computador A	Computador B	Computador C
Programa P1	1	10	20
Programa P2	1000	100	20
Tempo total (seg)	1001	110	40

- A é 10 vezes mais rápido que B para o programa P1
- B é 10 vezes mais rápido que A para o programa P2
- A é 20 vezes mais rápido que C para o programa P1
- C é 50 vezes mais rápido que A para o programa P2
- B é 2 vezes mais rápido que C para o programa P1
- C é 5 vezes mais rápido que B para o programa P2

Como resumir?

Tempo Total de Execução:

- B é 9.1 vezes mais rápido que A para os programas P1 e P2
- C é 25 vezes mais rápido que A para os programas P1 e P2
- C é 2.75 vezes mais rápido que B para os programas P1 e P2

Considera que cada um dos N programas do workload é executado o mesmo número de vezes.

Logo, média aritmética: $\text{tempo}_A = \frac{1}{N} \sum_{i=1}^{N} \text{tempo}_i$

Como resumir?

Se o programa P1 for muito mais freqüentemente executado na aplicação real que o programa P2?

Média ponderada: $\text{tempo}_A = \frac{1}{N} \sum_{i=1}^{N} P_i \times \text{tempo}_i$ e $\sum_{i=1}^{N} P_i = 1$

Considera que cada um dos N programas do workload é executado de acordo com sua Probabilidade de Ocorrência (P_i) ou Frequência Relativa

Exemplo Individual x Coletivo

	Máquinas			Ponderações		
programas	A	B	C	W(1)	W(2)	W(3)
P1	1	10	20	0.5	0.909	0.999
P2	1000	100	20	0.5	0.091	0.001

Média W(1)	500.5	55	20
Média W(2)	91.91	18.19	20
Média W(3)	2	10.09	20

Como resumir?

Tempo de Execução Normalizado. Nova abordagem para mix desigual de programas no workload.

Escolhe-se uma máquina para ser a Máquina Base. Tempo de Execução Normalizado do programa i executado na maquina A em relação à Maquina Base: $\bar{t_i} = \frac{\text{tempo\_A}_i}{\text{tempo\_Base}_i}$

Tempo de Execução Normalizado Médio: Média Geométrica dos tempos normalizados (usada no SPEC)

Razão_Média = $\sqrt[N]{\prod_{i=1}^{N} \bar{t_i}}$

Exemplo Individual x Coletivo

	Normalizado para A			Normalizado para B			Normalizado para C		
	A	B	C	A	B	C	A	B	C
P1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
P2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
Média Aritm.	1.0	5.05	10.01	5.05	1.0	1.1	25.03	2.75	1.0
Média Geom.	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0
Tempo Total	1.0	0.11	0.04	9.1	1.0	0.36	25.03	2.75	1.0

- Média Aritmética de tempos de execução normalizado não é consistente (depende da Máquina Base)
- Média Geométrica é consistente! Porém não reflete o tempo de execução.

A é 9.1 vezes mais lento que B; B é 2.75 vezes mais lento que C para workloads iguais

Fator importante para comparação

Reprodutibilidade: Fornecer subsídios para que outros consigam repetir o mesmo experimento validando o resultado apresentado.

Informações necessárias:

- Descrição detalhada do workload
- Descrição das diretivas de compilação
- Descrição do Sistema Operacional
- Configuração completa da máquina

Alguns frameworks para Benchmarking

Industry Standard (audited and verifiable):

- Standard Performance Evaluation Corporation (SPEC)
- Transaction Processing Performance Council (TPC)

Outros:

- HINT
- Fhourstones
- BAPCo
- Khornerstone
- Aquamark
- GL Excess
- John the Ripper
- The BRL-CAD Benchmark

Alguns frameworks para Benchmarking

Open Source benchmarks:

- Dhrystone: integer arithmetic performance
- Whetstone: floating-point arithmetic performance
- ApFloat: floating point
- Linpack / LAPACK
- GliBench: a Gui based benchmarking tool to check CPU and hard disk performance.
- BYTEmark benchmark suite
- STREAM: memory bandwidth benchmark
- MemPerf: memory bandwidth
- LLCBench: a group of benchmark for cache, MPI,etc.
- LMbench: a suite of simple, portable benchmarks
- nbench for Linux/Unix : Memory, integer and floating point comparison with AMD K6 233MHz
- Ubench - Unix Benchmark Utility
- NAS parallel benchmarks
- PovRay: 3D render
- Iozone file I/O
- Bonnie++: File I/O
- netperf : network throughput and latency benchmark
- GENESIS distributed memory benchmark suite
- Himeno Benchmark
- STREAM : measures sustainable memory bandwidth the corresponding computation rate for simple vector kernels.

Amdahl's Law

A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. It is a quantitative version of the **law of diminishing returns**.

$$\text{Tempo\_exec\_após\_melhoria} = \text{Tempo\_exec\_não\_afetado} + \frac{\text{Tempo\_exec\_afetado}}{\text{Quantidade\_melhoria}}$$

Otimização:

Suponha que um programa seja executado em 100 segundos em uma máquina, com multiplicação responsável por 80 segundos desse tempo. O quanto precisamos melhorar a velocidade da multiplicação se queremos que o programa seja executado 4 vezes mais rápido?

Que tal torná-lo 5 vezes mais rápido?

Conclusões

O desempenho é específico a um determinado programa: o tempo de execução total é um resumo consistente do desempenho.

Para uma determinada arquitetura, os aumentos de desempenho vêm de:

- aumentos na velocidade de clock (sem efeitos de CPI adversos)
- melhorias na organização do processador que diminuem a CPI
- melhorias no compilador que diminuem a CPI e/ou a contagem de instruções
- escolhas de algoritmo/linguagem que afetam a contagem de instruções

Embora tenhamos nos detido em análises de desempenho (Tempo), projetos reais **devem considerar custos** (além de diversos outros fatores: consumo, segurança, escalabilidade, etc).

Tradeoff típico de Engenharia: Projeto de Alto Desempenho x Projeto de Baixo Custo