Architektura procesora Intel Itanium (IA64)

Paweł Pisarczyk

Instytut Informatyki, Politechnika Warszawska

Pawel.Pisarczyk@ii.pw.edu.pl

Plan prezentacji

- Wprowadzenie
- Model programowy EPIC
- Mikroarchitektura procesora Itanium
- Wybrane mechanizmy procesora
- Porównanie wydajności procesorów IA32 i IA64
- Oprogramowanie
- Obliczeniowe klastry PC
- Podsumowanie

Kierunki rozwoju mikroprocesorów

- Zwiększenie prędkości wykonania instrukcji
- Przetwarzanie dużych zbiorów danych - SIMD
- Zwielokrotnienie ścieżek wykonania (np. Hyper Threading, wielordzeniowość)

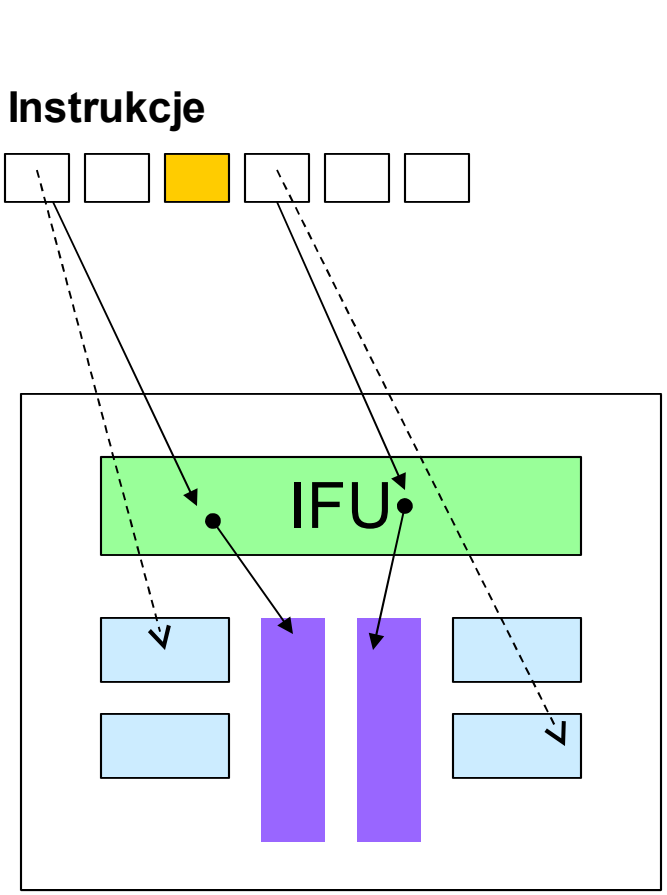
Rodzaje procesorów

- Procesory wektorowe
  - NEC SX-6
  - Procesory Cray (Cray X1)
- Procesory RISC
  - MIPS R8000
  - IBM Power4
  - Alpha 21164
- Procesory CISC
  - Intel Xeon (IA32)
  - AMD Opteron (x86-64)
- Procesory EPIC
  - Intel Itanium (IA64)

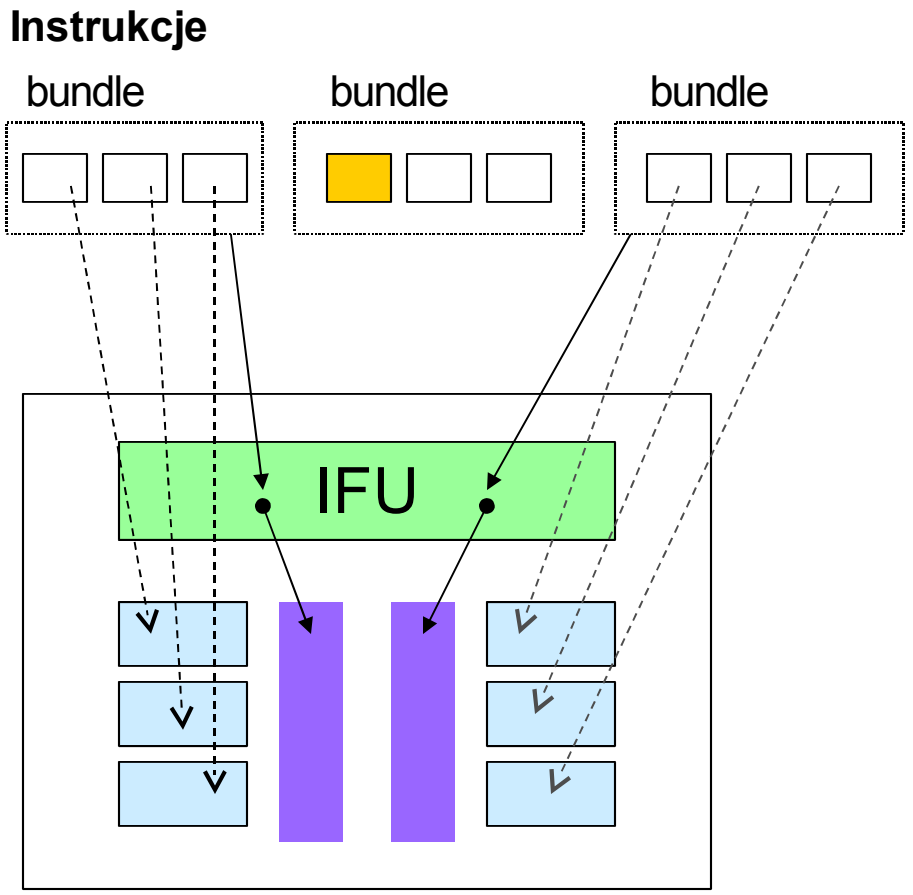
Architektura EPIC

- EPIC - Explicitly Parallel Instruction Computing
- Architektura wprowadzona w roku 2001 w procesorze Itanium (Merced)
- Zwielokrotnione jednostki wykonawcze
- Równoległe wykonanie instrukcji
- Duża liczba rejestrów, zestaw prostych, ortogonalnych instrukcji
- Ciężar optymalizacji wykonania przeniesiony na kompilator

RISC, CISC vs EPIC



RISC lub CISC

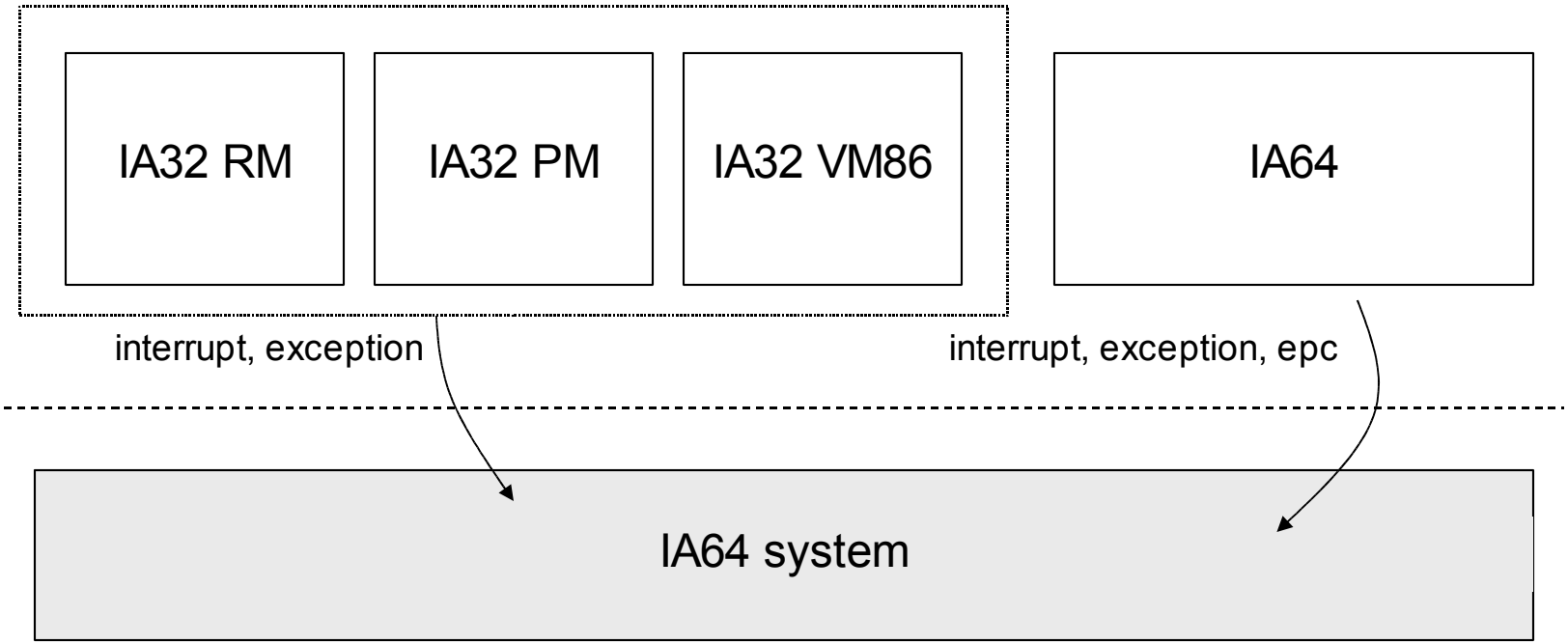


EPIC

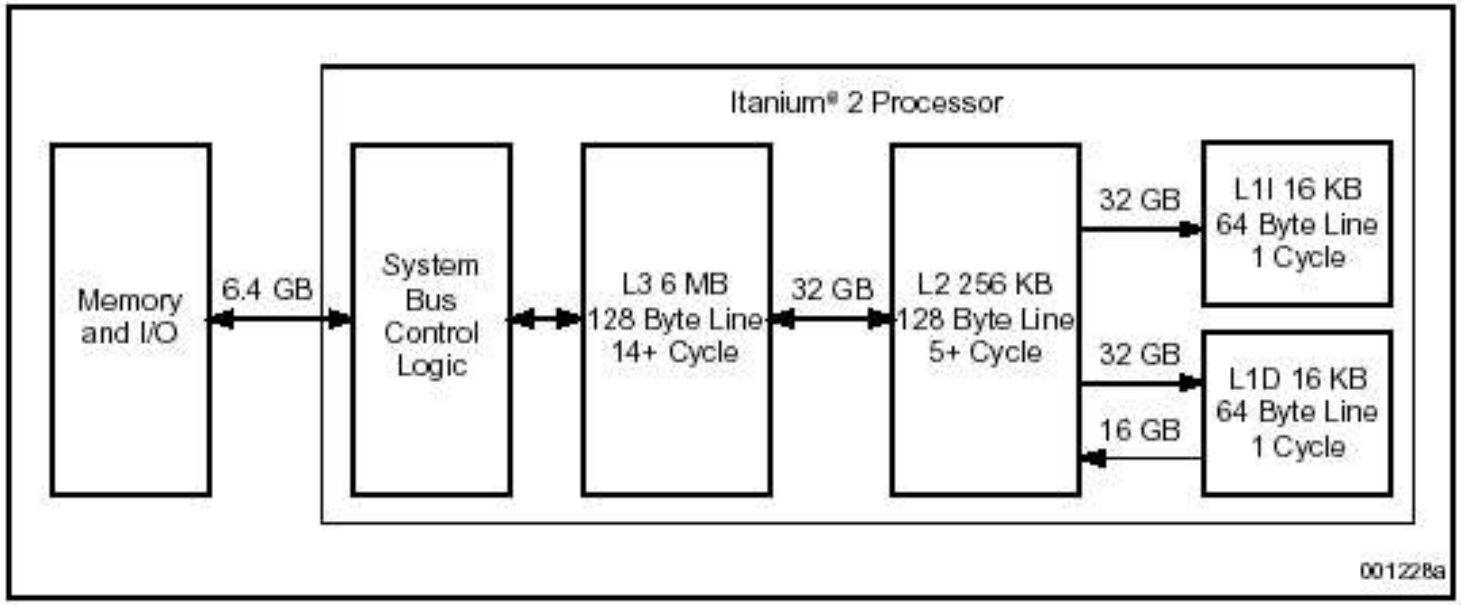
Procesor Itanium2

- Architektura 64-o bitowa typu EPIC - VLIW, nowy model programowy - IA64
- Wykonanie do sześciu instrukcji w jednym cyklu zegara
- Silnie zwielokrotnione jednostki wykonawcze
- Bardzo duża liczba rejestrów roboczych
- Mechanizmy zwiększające efektywność wykonania kodu
- Możliwość wykonania kodu IA32

Środowisko wykonania



Hierarchia pamięci cache



Jednostki funkcjonalne

- jednostka DCU (Data Cache Unit): 4 porty pamięci - 2 porty typu store, 2 porty typu load
- arytmetyka stałopozycyjna: 6 jednostek ALU0-5, jednostka ISHIFT
- arytmetyka zmiennopozycyjna: 2 jednostki FMAC (dodawanie-mnożenie), 2 jednostki FMISC (pozostałe operacje)
- instrukcje multimedialne: 6 jednostek wektorowych PALU0-5, 2 jednostki shift PSMU0-1, jednostka mnożenia równoległego PMUL, jednostka zliczająca POPCNT
- skoki: 3 jednostki skoków

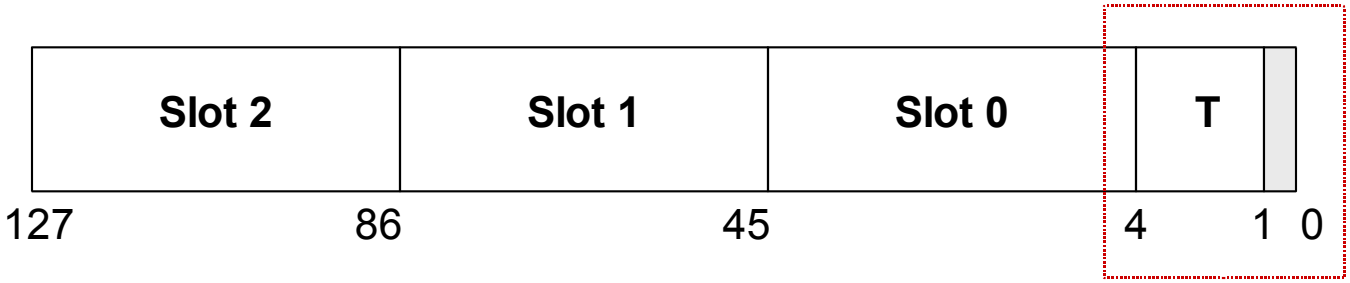
Rejestry aplikacyjne

- 128 rejestrów ogólnego przeznaczenia: gr0-127
- 128 rejestrów zmiennopozycyjnych: fr0-127
- 64 rejestry predykatów: pr0-63
- 8 rejestrów skoków: br0-br7
- 128 rejestrów aplikacyjnych: ar0-ar127
- rejestry CPUID
- rejestry danych PMU
- rejestr IP, CFM i UM

Rejestry systemowe

- 8 rejestrów regionów: rr0-rr7
- Minimalnie 16 rejestrów kluczy ochrony: pkr0-pkr*n*
- Rejestry TLB: itr0-itr*n*, itc0-itc*n*, dtr0-dtc*n*
- Rejestry uruchomieniowe: ibr0-ibr*n*, dbr0-dbr*n*
- Rejestry konfiguracyjne PMU: pmc0-pmc*n*
- Rejestry sterujące: cr0-cr*n*
- Rejestr PSR (Processor Status Register)

Format instrukcji



Przykład kodu maszynowego

```
{ .mfi
add r32=1,r33
fcvt.xf f39=f35
nop.i 0
}

{ .mfi
nop.m 0
fma.d.s1 f36=f41,f1,f0
nop.i 0 ;;
}

{ .mfi
nop.m 0
(p23) fcmp.gt.s0 p22,p0=f8,f38
nop.i 0
}

{ .mmb
nop.m 0
setf.sig f32=r32
(p23) br.wtop.dptk .b1_3 ;;
}
```

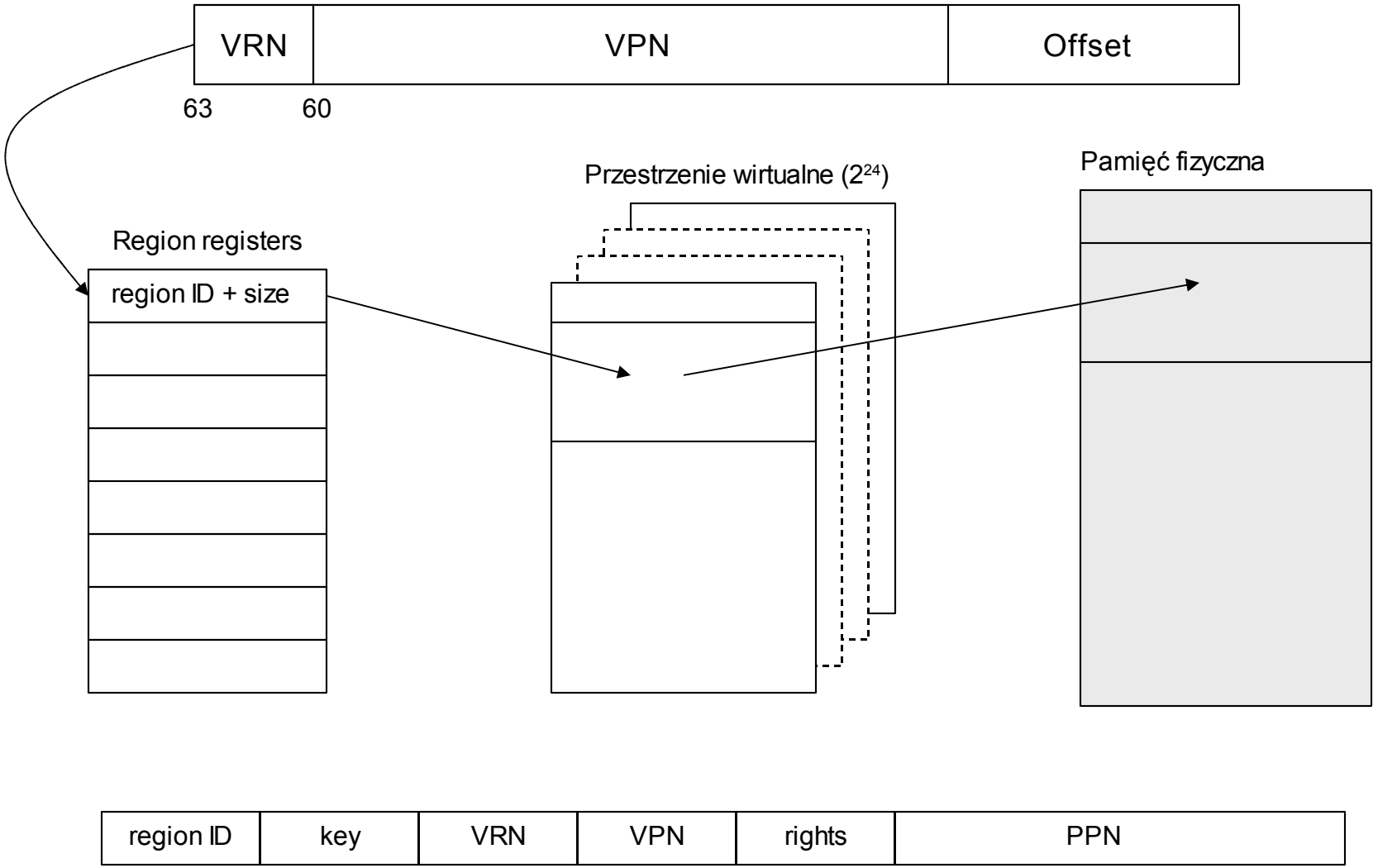
Mechanizmy ochrony

- Cztery poziomy ochrony 0-3 (poziom 0 najbardziej uprzywilejowany)
- Rejestry systemowe dostępne są wyłącznie na poziomie 0
- Ochrona oparta o klucze i domeny ochrony + atrybuty stron
- Przejścia między poziomami ochrony po wystąpieniu przerwania (wyjątku) lub za pośrednictwem instrukcji `epc` i stron typu prominent
- Oddzielny atrybut wykonywalności strony

Zarządzanie pamięcią

- Dwa tryby adresowania: adresowanie fizyczne i wirtualne
- Pamięć wirtualna oparta o stronicowanie i regiony pamięci
- Możliwość realizacji modelu pamięci MAS lub SAS
- Płaska, liniowa przestrzeń adresowa
- 63-bitowy adres fizyczny
- Strony wielkości od 4 KB do 4 GB
- Zestaw rejestrów sterujących TLB
- Tablica stron stanowi rozszerzenie TLB
- Brak konieczności wymiatania TLB

Stronicowanie



Rejestry predykatów

```
if (r1)
        r2 = r3 + r4;
else
        r7 = r6 - r5;
```

```
cmp.ne p1,p2 = r1, 0;;
[p1] add r2 = r3, r4
[p2] sub r7 = r6, r5
```

Wywołanie funkcji IA32

```
test(a, b);

push eax      ; b
push edx      ; a
call test
add esp, 8
```

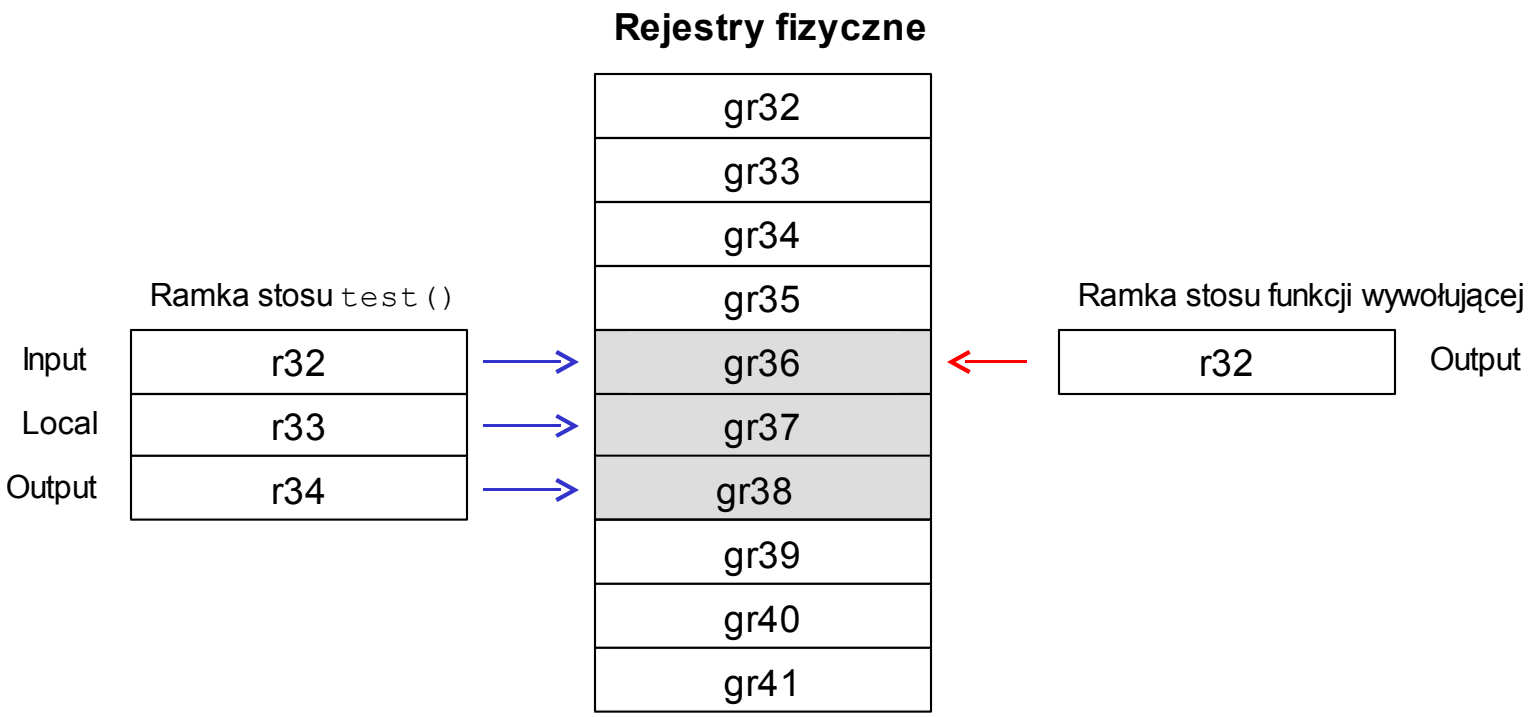
```
int test(int a, int b)
{
  int c = a;
  return c;
}

push ebp
mov ebp, esp
sub esp, 4h

mov eax, dword ptr [ebp + 8]
mov dword ptr [ebp - 4], eax
mov eax, dword ptr [ebp - 4]

mov esp, ebp
pop ebp
ret
```

Wywołanie funkcji IA64 (RSE)

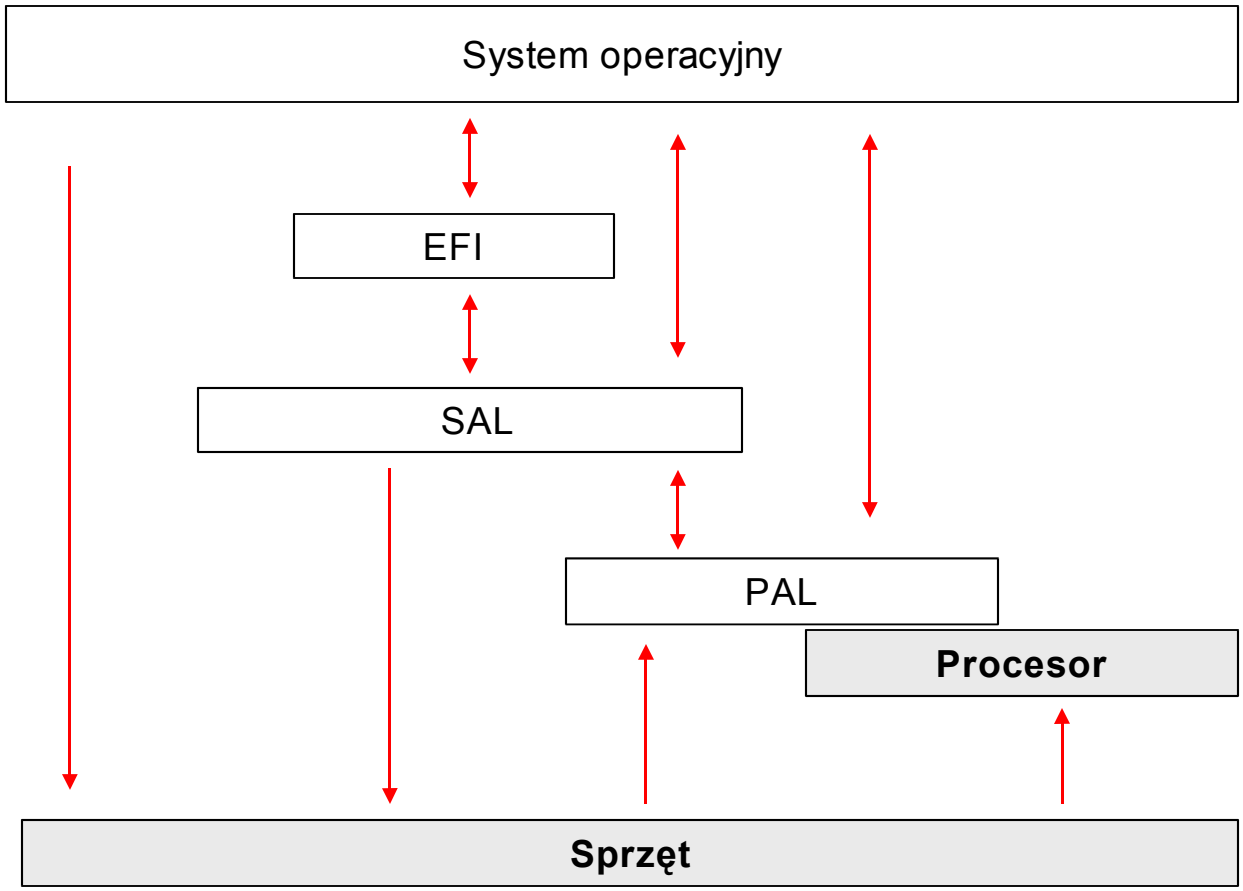
```
test(a);

// Call test(a), 'a' is in R32
1: (p0) br.call b0=test
```

```
int test(int a)
{
  int c = a;
  return c;
}

2: .proc
3: test:

// Set up register stack
4: alloc r4=ar.pfs,1,1,1,2

// int c = a
5: (p0) mov r33=r32
6: ;;

// return c
7: (p0) mov r34=r33
8: (p0) mov ar.pfs=r4
9: (p0) br.ret.sptk.few b0
10: .endp
```

Mapowanie rejestrów (RSE)



PAL, SAL, EFI



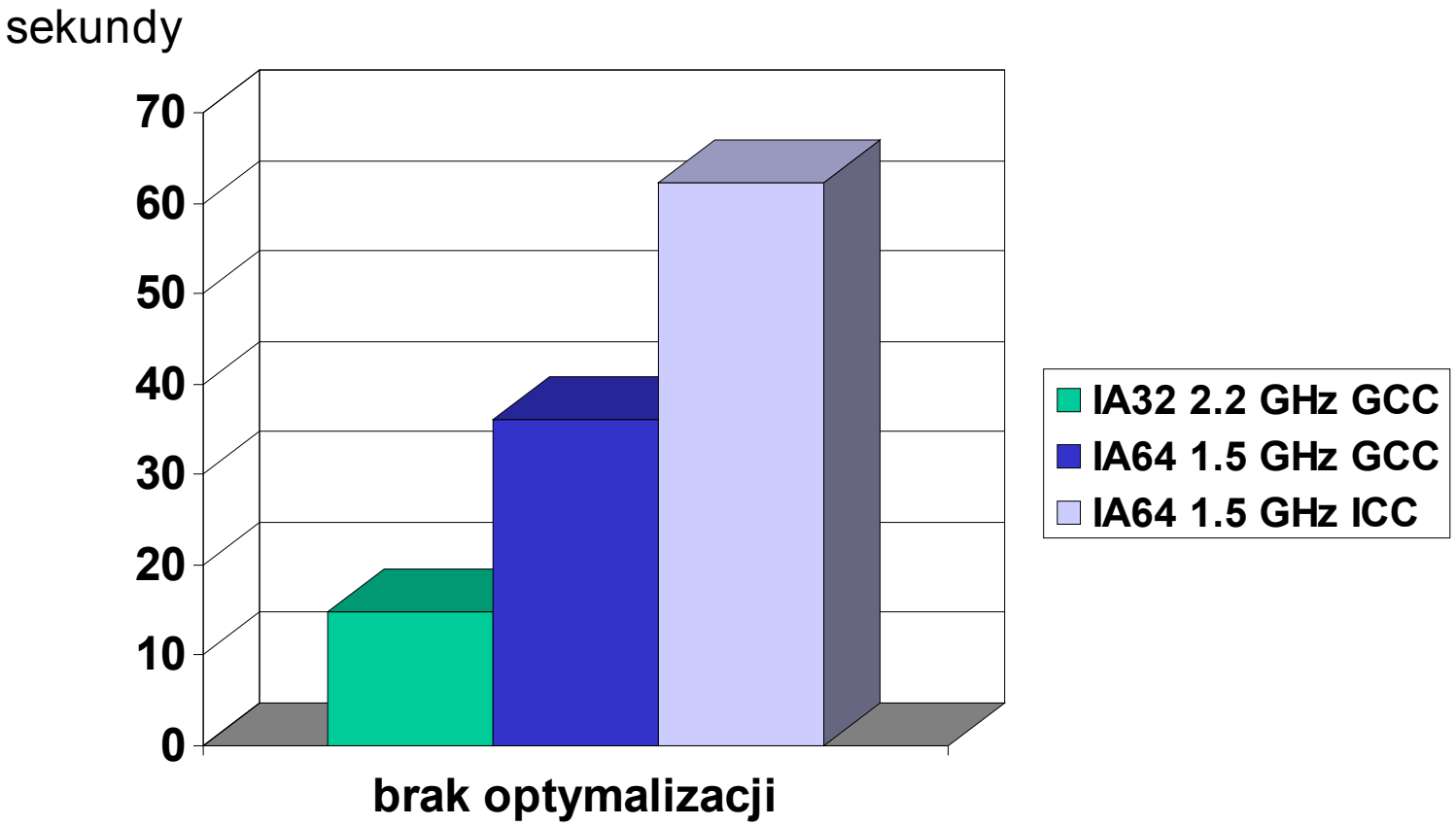
Testy IA64

- Testy przy pomocy istniejących aplikacji (Povray, Scilab, John the Ripper etc.)
- Opracowanie prostego testowego programu, uniemożliwiającego stosowanie powszechnych metod optymalizacji (pomijanie obliczeń, rozwijanie pętli itp.)
- Kompilacja przy pomocy GNU CC i Intel ICC

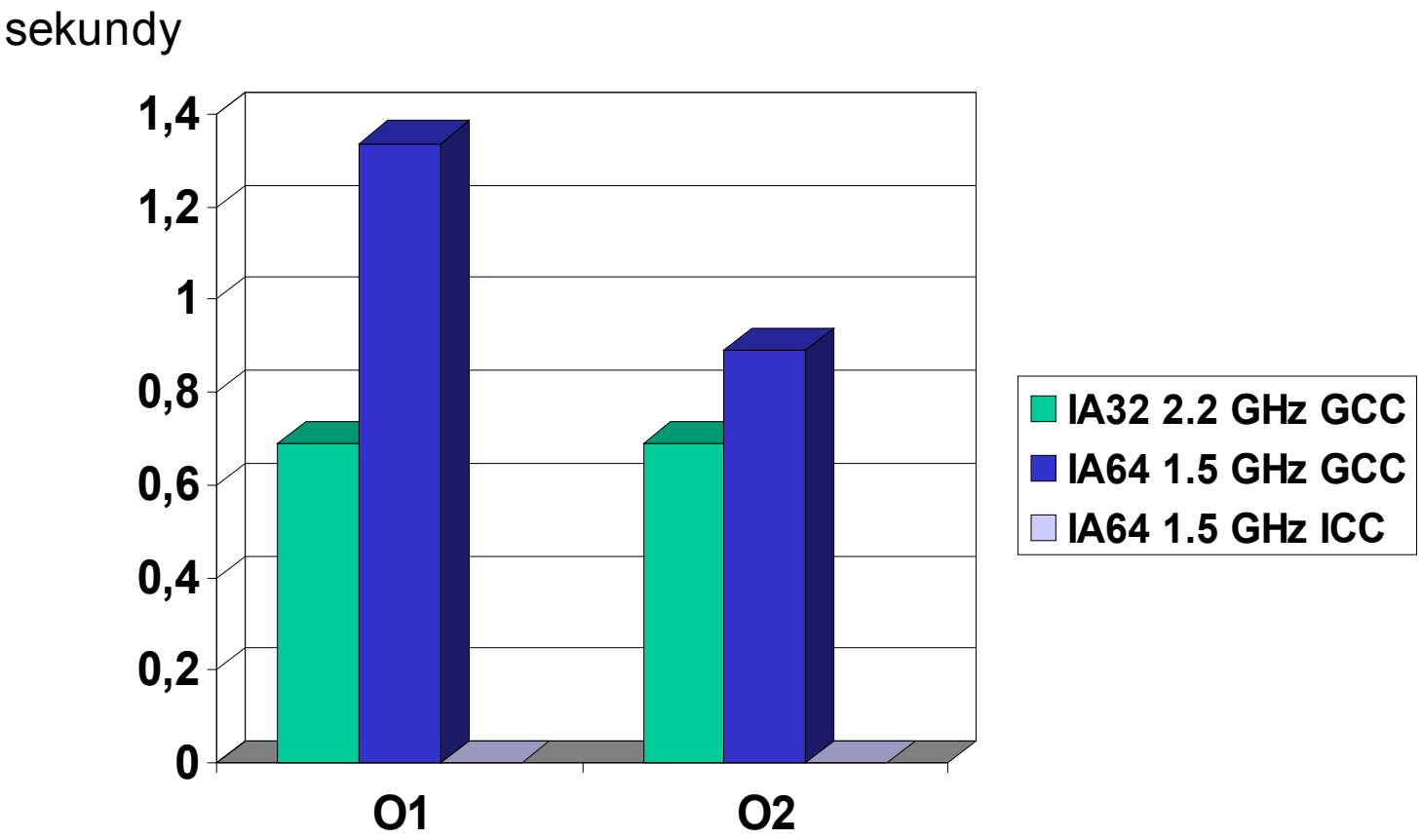
Kod testowy

```
...
...
for (k = 0; k < 1 000 000 000; k++) {
     v *= k / A + k * B - C;
}
...
...
return v;
```

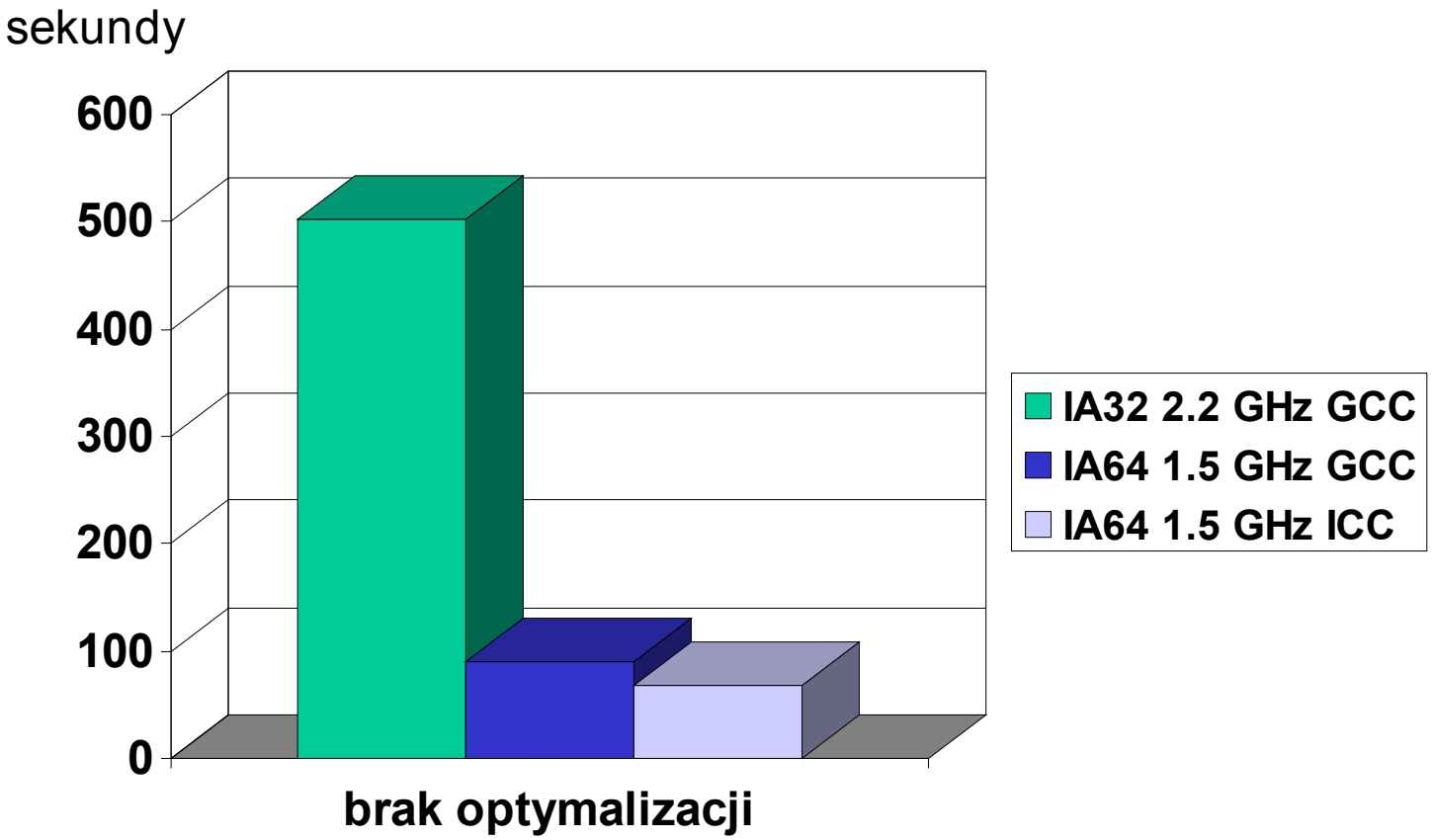
Wyniki (czas obliczeń INT)



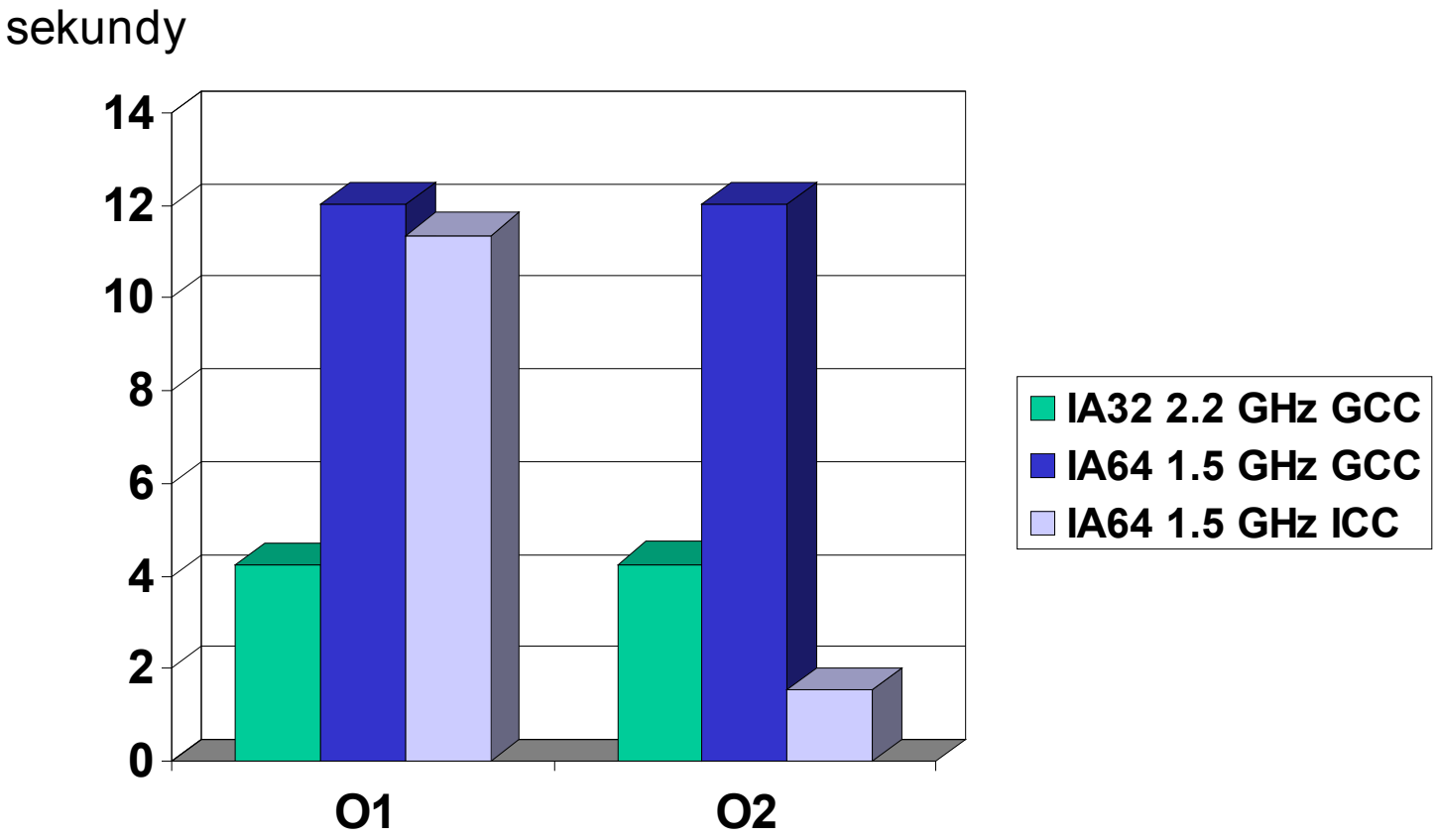
Wyniki (czas obliczeń INT)



Wyniki (czas obliczeń FLOAT)



Wyniki (czas obliczeń FLOAT)



Oprogramowanie

- Systemy operacyjne: MS Windows 2003, GNU/Linux, HP-UX
- Kompilatory: GNU CC, Intel ICC, G77, Intel Fortran itp.
- Narzędzia do profilowania kodu i monitorowania wydajności: VTUNE, Vampire
- Biblioteki numeryczne: HP MLIB, Intel MKL, Intel IPP, NAG
- Narzędzia OpenSource stworzone w ramach Gelato (np. perfmon i pm-utils)

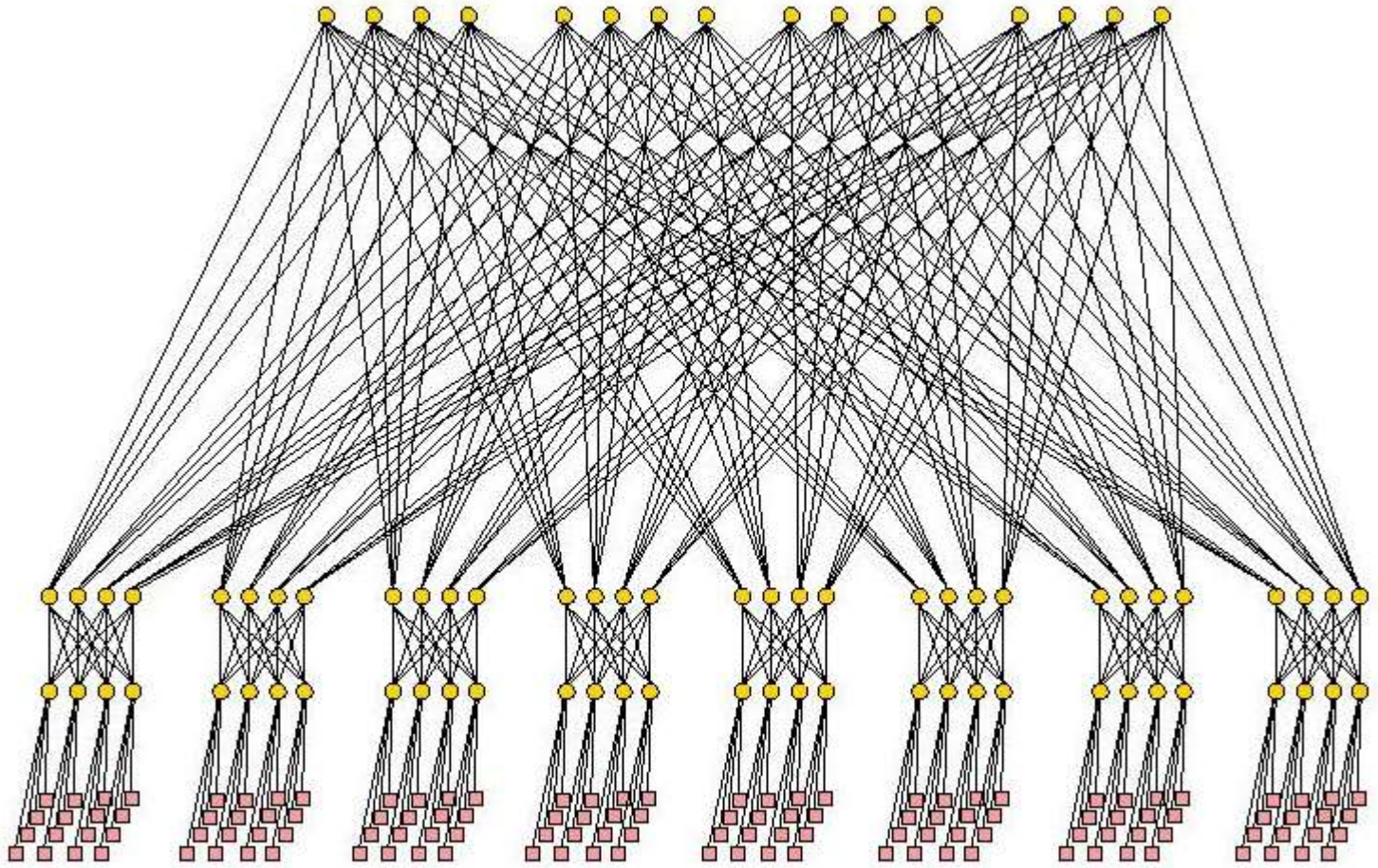
Obliczeniowe klastry PC

- Wykorzystanie popularnych technologii (np. PC IA32)
- Węzły połączone przy pomocy sieci Ethernet, Myrinet, QsNET
- Popularne systemy operacyjne (np. Linux, BSD, Windows)
- Do prowadzenia obliczeń wykorzystywane są specjalne środowiska programowe (MOSIX, Beowulf, Quadrics RMS, PVM) i biblioteki (MPI, PVM)

Cechy połączeń międzywęzłowych

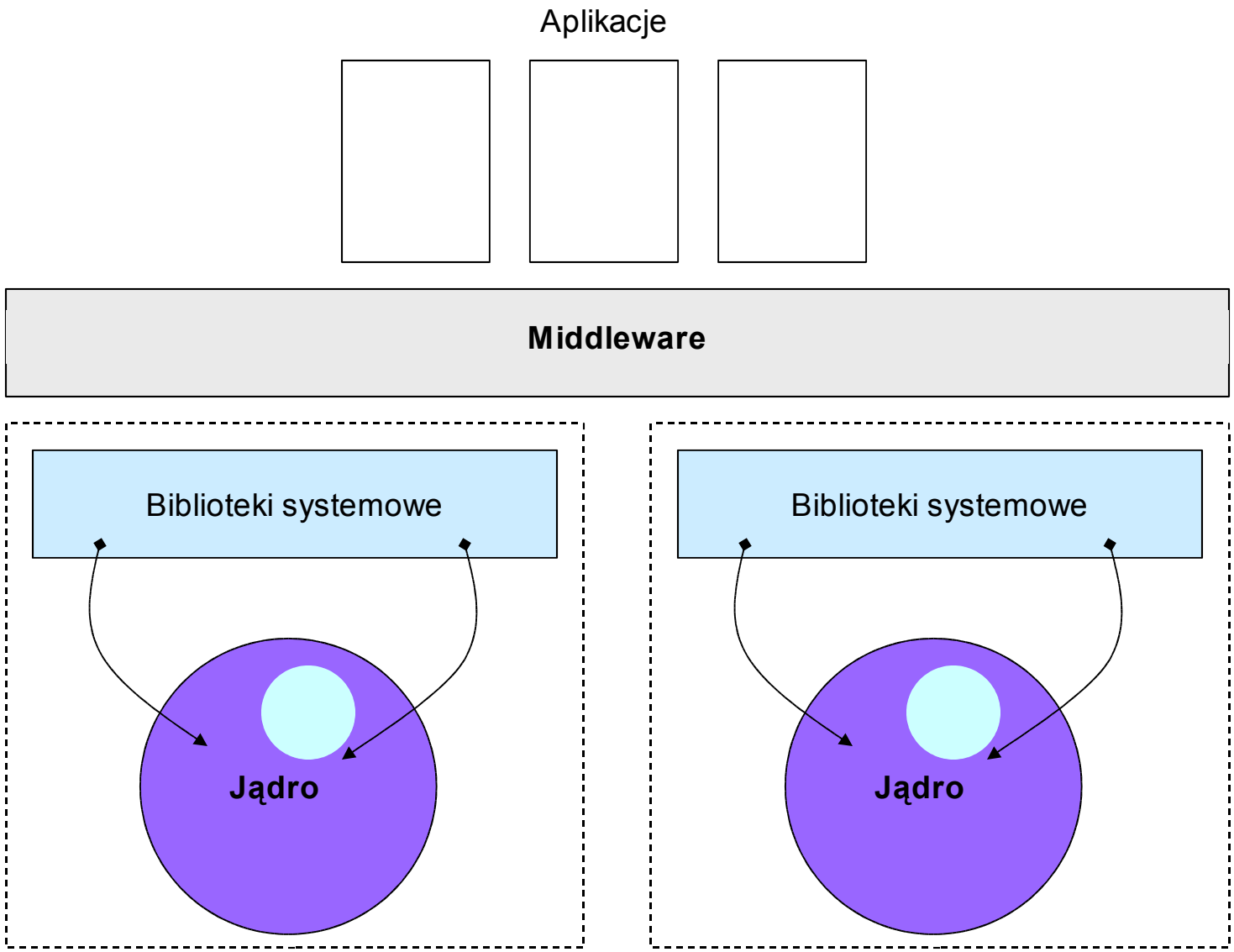
- Minimalizacja opóźnień i czasu przekazywania komunikatów
- Specjalizowana topologia połączeń (hiperkostka, torus, hierarchiczne drzewo)
- Uporządkowany sposób dostępu do medium
- Duża przepływność kanału komunikacyjnego (2GB/s)
- Pełne wykorzystanie pasma magistrali lokalnej, odciążenie procesora głównego
- Przykłady sieci - Myrinet, InfiniBand, Quadrics QsNET (ELAN2)

Topologia połączeń Fat Tree



Rysunek z „Overview of Recent Supercomputers 2003”
Aad J. van der Steen, Jack J. Dongarra, http://www.top500.org

Klastry


Aplikacje
Middleware
Biblioteki systemowe
Jądro
Biblioteki systemowe
Jądro

Przykłady rozwiązań - klastry

Linux Networx/Quadrics

- Procesor: Intel Xeon 2.4 GHz
- Liczba procesorów: 2304
- Komunikacja: QsNet ELAN3
- System operacyjny: Linux + RMS + Lustre
- Maksymalna wydajność: 7.6 TFLOPS
- 3 komputer w rankingu TOP500

- Zbudowany dla LLNL w 2002 roku

Podsumowanie

- Itanium wymaga odpowiednich kompilatorów oraz narzędzi do monitorowania wydajności i profilowania kodu - projekt Gelato (HP Labs, CERN, NCSA, ...)
- Trzykrotnie efektywniejszy model programowy w stosunku do IA32
- Nowy model programowy IA64, pozbawiony ograniczeń IA32
- Odejście od przestarzałej architektury PC
- Istnieje możliwość przeniesienia systemu Phoenix na IA64 w celu szczegółowego poznania nowej architektury i walidacji systemu