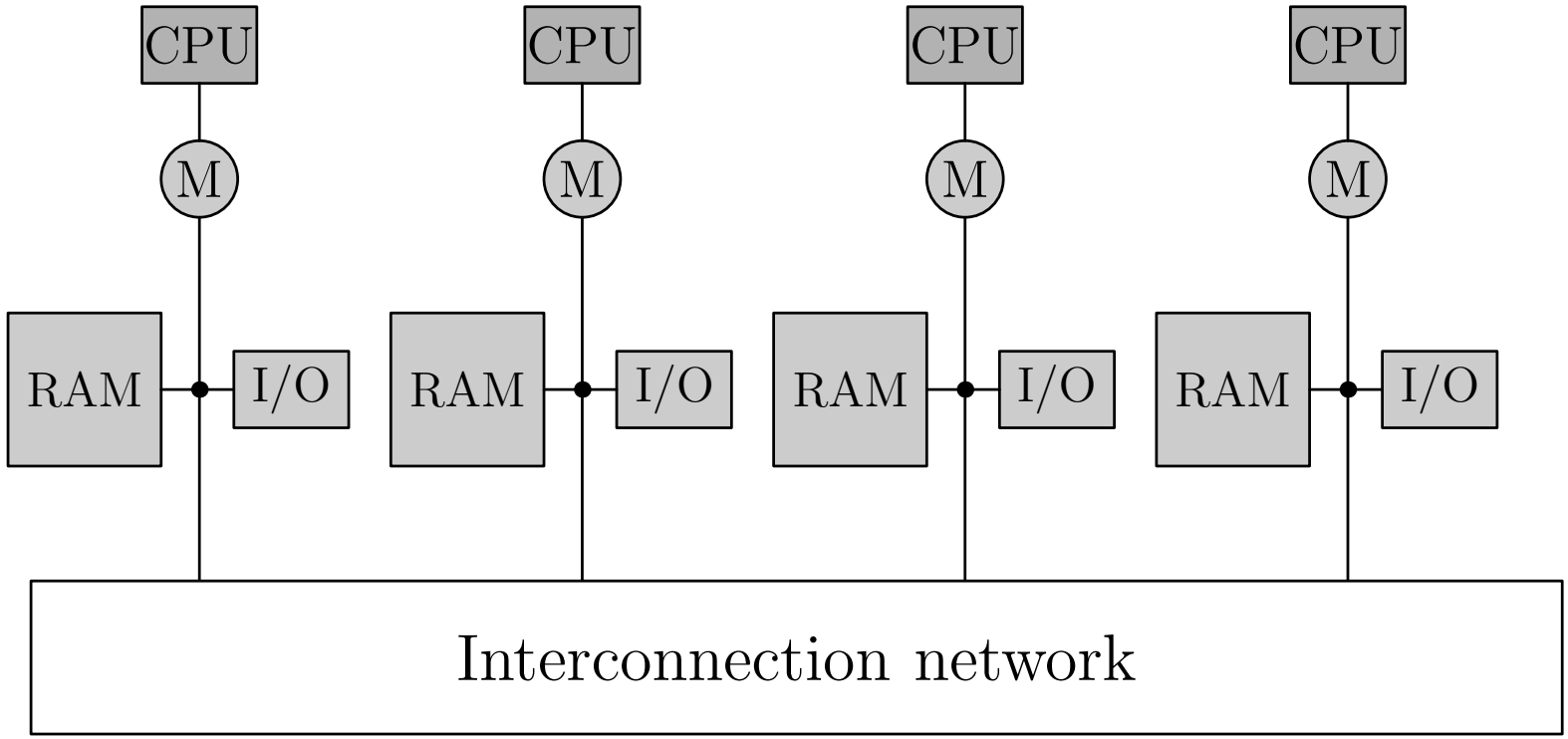
Paralelní architektury se sdílenou pamětí typu NUMA

NUMA architektury

Multiprocesorové systémy s distribuovanou pamětí I.

- úzkým hrdlem multiprocesorů se sdílenou pamětí je datová komunikace
- s rostoucím počtem procesorů není paměťový systém schopen obsluhovat všechny požadavky
- problem *cache coherence* se také stává komplikovanější
- může se stát, že přidáváním dalších procesorů výkon multiprocesoru dokonce sníží
- počet CPU u multiprocesorů se sdílenou pamětí je omezen na několik desítek ≈ 64
- řešením je zavedení multiprocesorů s distribuovanou pamětí
  - příklad - Cray T3D, 1993

Multiprocesorové systémy s distribuovanou pamětí II.



Multiprocesorové systémy s distribuovanou pamětí III.

- každý procesor má rychlý přístup do své paměti, ale velice pomalý přístup do cizí paměti
- předpokládá se, že běžný kód splňuje *spatial locality*
- většina přístupů do paměti tedy bude probíhat na úrovni lokální paměti daného CPU
- což snižuje zátěž na IN - *interconnection*
- a to umožňuje vytvářet systémy s mnohem větším počtem CPU (řádově tisíce)
- soubor všech lokálních pamětí jednotlivých CPU dohromady tvoří jeden logický *adresový prostor*
- protože rychlost přístupu k jednotlivým adresám se liší, mluvíme o **NUMA** architektuře
  - **NUMA** = non-uniform memory acces
  - rozdíl může být až stonásobný

Multiprocesorové systémy s distribuovanou pamětí IV.

- problém *cache coherence* nemůže být řešen pomocí "čmuchacího"protokolu
  - multiprocesory s distribuovanou pamětí nepoužívají sběrnici
  - ani s jiném komunikačním zařízením by to s tak velkým počtem procesorů nebylo realizovatelné
- místo toho se používá tzv. adresář $\rightarrow$ **directory-based protocol**

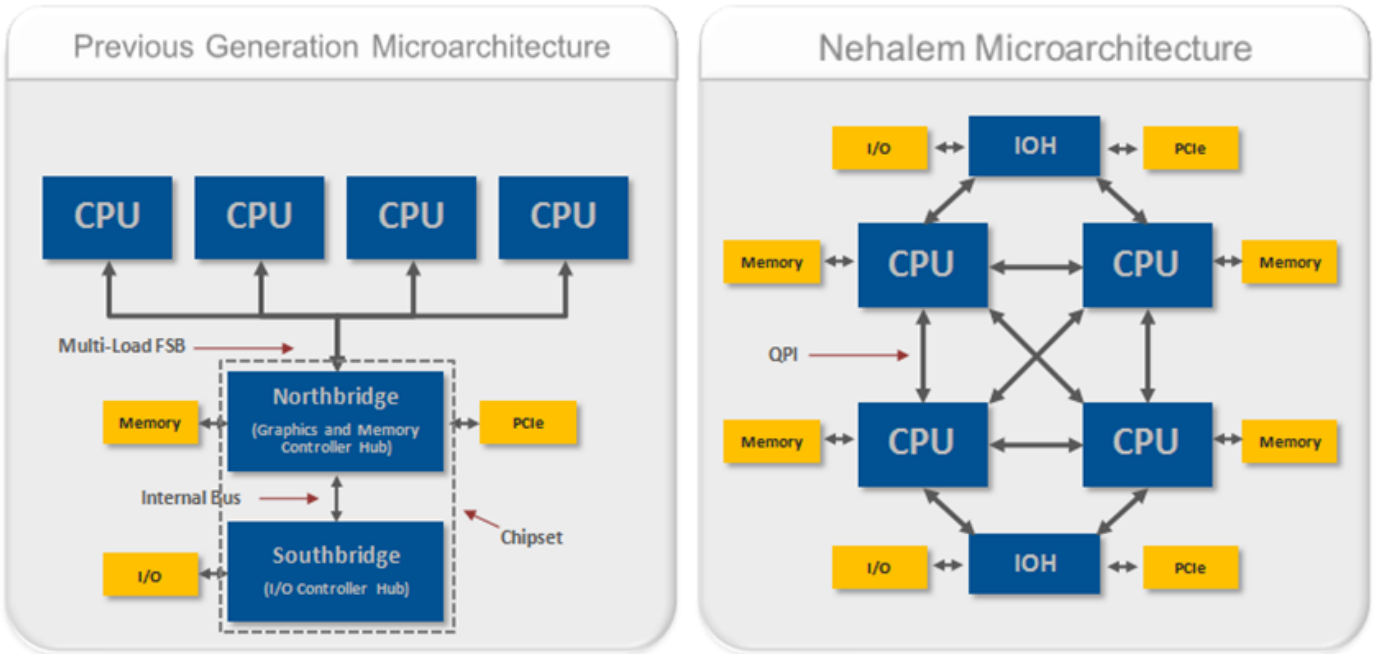
Multiprocesorové systémy s distribuovanou pamětí V.

Directory based systems

- jde o systémy doplněné tabulkou popisující stavy jednotlivých bloků paměti
- ty mohou být opět *shared, invalid* a *modified*
  - *SHARED* - proměnná je sdílena více CPU, ale všichni ji používají jen ke čtení - její hodnota v globální virtuální paměti je korektní
  - *INVALID* - jeden z procesorů provedl zápis - hodnota proměnné v globální virtuální paměti není správná
  - *MODIFIED* - procesor, který má proměnnou označenou za *modified*, je ten, který k ní má momentálně exkluzivní přístup
- tento adresář může být velký a zásadně zpomalovat přístup do paměti
- někdy se provádí jeho rozdělení mezi všechny procesory, čímž se zátěž distribuuje na celý systém

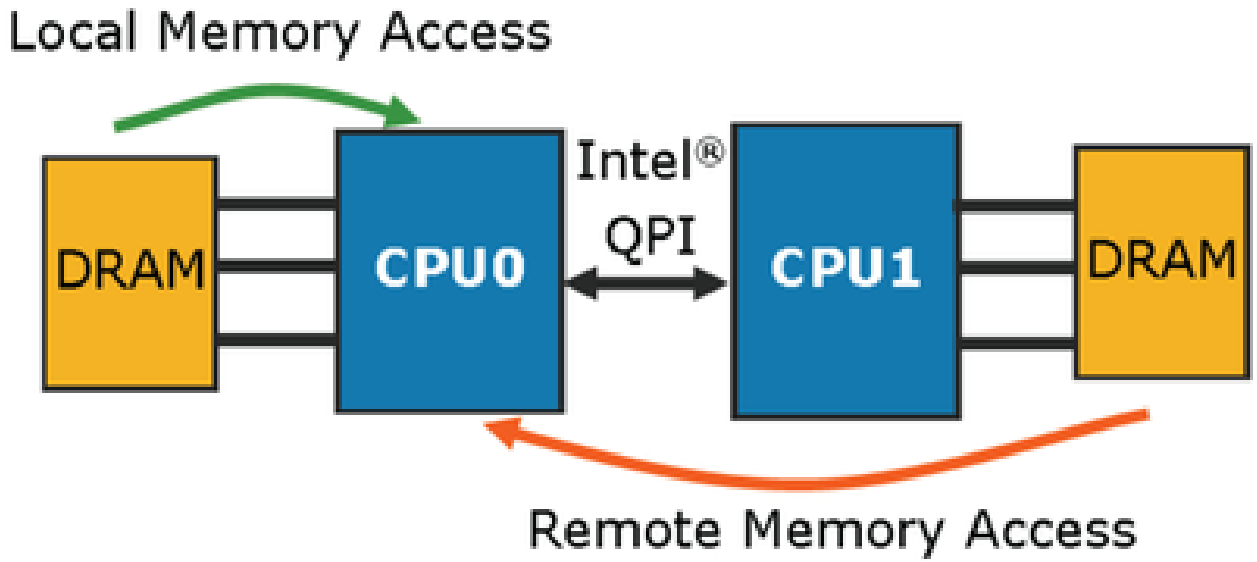
Intel Quick Path Interconnect

Intel Nehalem Quick Path Interconnect



Intel Quick Path Interconnect

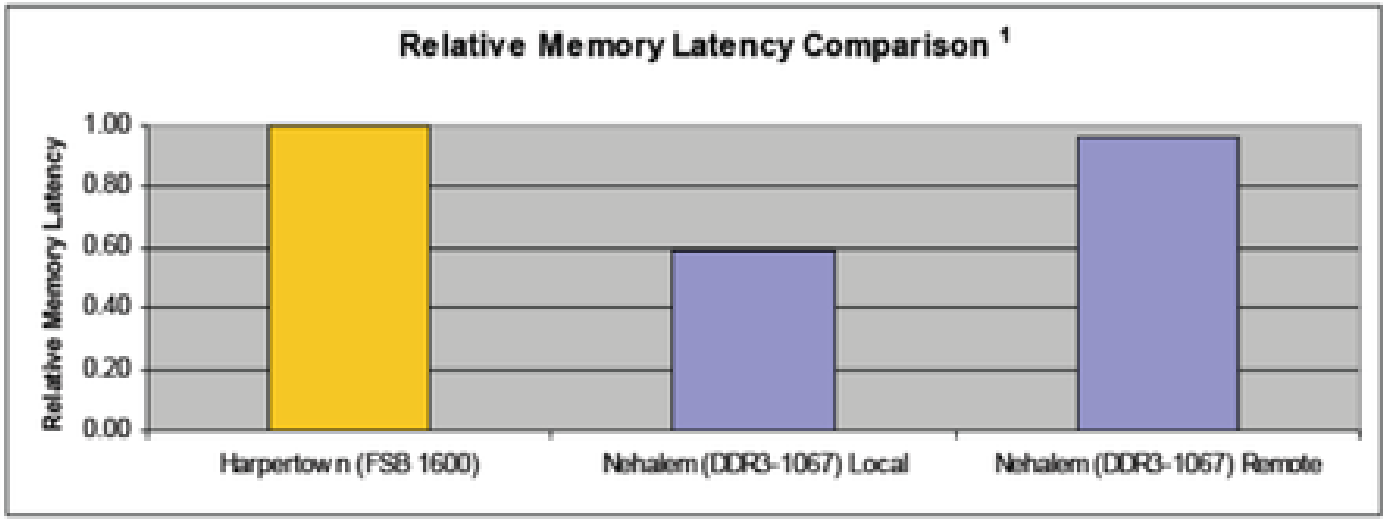
Intel Nehalem Quick Path Interconnect



Zdroj: `http://www.tomshardware.com/reviews/Intel-i7-nehalem-cpu,2041.html`

Intel Quick Path Interconnect

Intel Nehalem Quick Path Interconnect



Zdroj: `http://www.tomshardware.com/reviews/Intel-i7-nehalem-cpu,2041.html`

Intel Quick Path Interconnect

- na současných architekturách je zpomalení při přístupu do vzdáleného paměťového čipu řádově desítky procent (cca. 50%)
- v ideálním případě by každé vlákno mělo přistupovat do svého paměťového modulu
- v nejhorším případě budou všechna vlákna přistupovat jen do jednoho modulu
  - vše bude obsluhovat jeden paměťový řadič a ostatní zůstanou nevyužity
- během výpočtu navíc vlákna migrují mezi různými jádry

Migrace vláken

Příklad: Migraci vláken demonstruje následující program.

```
1  #include <omp.h>
2  #include <iostream>
3  #include <sched.h>
4  #include <time.h>
5
6  using namespace std;
7
8  int main( int argc, char* argv[] )
9  {
10    cout << "Press␣Ctrl-C␣to␣quit." << endl;
11 #pragma omp parallel
12    {
13       int core( -1 );
14       while( true )
15       {
16          int currentCore = sched_getcpu();
17          if( currentCore != core )
18          {
19             time_t rawtime;
20             struct tm * timeinfo;
21             time ( &rawtime );
22             timeinfo = localtime ( &rawtime );
23             cout << "Thread␣" << omp_get_thread_num()
24                  << "␣is␣migrating␣to␣the␣core␣"
25                  << currentCore << "␣at␣" << asctime (timeinfo);
26             core = currentCore;
27          }
28       }
29    }
30 }
```

Migrace vláken

Takto může vypadat výstup programu:

```
1  Press Ctrl-C to quit.
2  Press Ctrl-C to quit.
3  Thread 5 is migrating to the core 5 after 386991 CPU ticks at Sat Mar 14 15:08:58 2015
4  Thread 3 is migrating to the core 0 after 734313 CPU ticks at Sat Mar 14 15:08:58 2015
5  Thread 1 is migrating to the core 4 after 803332 CPU ticks at Sat Mar 14 15:08:58 2015
6  Thread 2 is migrating to the core 1 after 1175374 CPU ticks at Sat Mar 14 15:08:58 2015
7  Thread 3 is migrating to the core 3 after 1289344 CPU ticks at Sat Mar 14 15:08:58 2015
8  Thread 0 is migrating to the core 4 after 1213527 CPU ticks at Sat Mar 14 15:08:58 2015
9  Thread 4 is migrating to the core 1 after 12916 CPU ticks at Sat Mar 14 15:08:58 2015
10 Thread 0 is migrating to the core 1 after 73309 CPU ticks at Sat Mar 14 15:08:58 2015
11 Thread 0 is migrating to the core 3 after 16392 CPU ticks at Sat Mar 14 15:08:58 2015
12 Thread 2 is migrating to the core 3 after 16516469 CPU ticks at Sat Mar 14 15:08:58 2015
13 Thread 2 is migrating to the core 2 after 26534 CPU ticks at Sat Mar 14 15:08:58 2015
14 Thread 3 is migrating to the core 2 after 4435367910 CPU ticks at Sat Mar 14 15:09:00 2015
15 Thread 3 is migrating to the core 0 after 33729 CPU ticks at Sat Mar 14 15:09:00 2015
16 Thread 3 is migrating to the core 3 after 590357448 CPU ticks at Sat Mar 14 15:09:00 2015
17 Thread 0 is migrating to the core 0 after 5061599111 CPU ticks at Sat Mar 14 15:09:00 2015
18 Thread 0 is migrating to the core 5 after 735083041 CPU ticks at Sat Mar 14 15:09:00 2015
19 Thread 5 is migrating to the core 0 after 5819493402 CPU ticks at Sat Mar 14 15:09:00 2015
20 Thread 5 is migrating to the core 4 after 3052310067 CPU ticks at Sat Mar 14 15:09:01 2015
21 Thread 5 is migrating to the core 0 after 32086 CPU ticks at Sat Mar 14 15:09:01 2015
22 Thread 4 is migrating to the core 4 after 15851139081 CPU ticks at Sat Mar 14 15:09:04 2015
23 Thread 1 is migrating to the core 1 after 15853509961 CPU ticks at Sat Mar 14 15:09:04 2015
24 Thread 5 is migrating to the core 2 after 12766345638 CPU ticks at Sat Mar 14 15:09:06 2015
25 Thread 2 is migrating to the core 0 after 21705683629 CPU ticks at Sat Mar 14 15:09:06 2015
26 Thread 2 is migrating to the core 1 after 711091745 CPU ticks at Sat Mar 14 15:09:06 2015
27 Thread 1 is migrating to the core 0 after 6588260892 CPU ticks at Sat Mar 14 15:09:06 2015
28 Thread 2 is migrating to the core 0 after 5461163460 CPU ticks at Sat Mar 14 15:09:08 2015
29 Thread 1 is migrating to the core 3 after 5561530993 CPU ticks at Sat Mar 14 15:09:08 2015
30 Thread 3 is migrating to the core 4 after 23001697404 CPU ticks at Sat Mar 14 15:09:08 2015
31 Thread 3 is migrating to the core 1 after 84379524 CPU ticks at Sat Mar 14 15:09:08 2015
32 Thread 2 is migrating to the core 2 after 502115392 CPU ticks at Sat Mar 14 15:09:08 2015
33 Thread 2 is migrating to the core 0 after 9156 CPU ticks at Sat Mar 14 15:09:08 2015
34 Thread 3 is migrating to the core 0 after 305147445 CPU ticks at Sat Mar 14 15:09:08 2015
35 Thread 3 is migrating to the core 1 after 115674659 CPU ticks at Sat Mar 14 15:09:08 2015
36 Thread 3 is migrating to the core 5 after 385608623 CPU ticks at Sat Mar 14 15:09:08 2015
37 Thread 3 is migrating to the core 1 after 53204 CPU ticks at Sat Mar 14 15:09:08 2015
38 Thread 3 is migrating to the core 3 after 189602415 CPU ticks at Sat Mar 14 15:09:08 2015
39 Thread 3 is migrating to the core 4 after 19276581 CPU ticks at Sat Mar 14 15:09:08 2015
40 Thread 4 is migrating to the core 1 after 13335348861 CPU ticks at Sat Mar 14 15:09:08 2015
41 Thread 4 is migrating to the core 5 after 528286578 CPU ticks at Sat Mar 14 15:09:08 2015
42 Thread 0 is migrating to the core 1 after 24026530917 CPU ticks at Sat Mar 14 15:09:08 2015
43 Thread 0 is migrating to the core 5 after 96248748 CPU ticks at Sat Mar 14 15:09:08 2015
44 Thread 4 is migrating to the core 4 after 220570368 CPU ticks at Sat Mar 14 15:09:08 2015
45 Thread 3 is migrating to the core 1 after 799437134 CPU ticks at Sat Mar 14 15:09:08 2015
46 Thread 3 is migrating to the core 5 after 116768426 CPU ticks at Sat Mar 14 15:09:08 2015
47 Thread 0 is migrating to the core 1 after 349676495 CPU ticks at Sat Mar 14 15:09:08 2015
48 Thread 3 is migrating to the core 3 after 580984971 CPU ticks at Sat Mar 14 15:09:09 2015
49 Thread 3 is migrating to the core 5 after 48299802 CPU ticks at Sat Mar 14 15:09:09 2015
50 Thread 2 is migrating to the core 4 after 2865194039 CPU ticks at Sat Mar 14 15:09:09 2015
51 Thread 4 is migrating to the core 0 after 1385208571 CPU ticks at Sat Mar 14 15:09:09 2015
52 Thread 4 is migrating to the core 5 after 157697835 CPU ticks at Sat Mar 14 15:09:09 2015
53 Thread 1 is migrating to the core 0 after 3544254891 CPU ticks at Sat Mar 14 15:09:09 2015
54 Thread 4 is migrating to the core 3 after 51787053 CPU ticks at Sat Mar 14 15:09:09 2015
55 Thread 1 is migrating to the core 5 after 361267847 CPU ticks at Sat Mar 14 15:09:09 2015
56 Thread 3 is migrating to the core 0 after 1227173431 CPU ticks at Sat Mar 14 15:09:09 2015
57 Thread 1 is migrating to the core 1 after 1265811160 CPU ticks at Sat Mar 14 15:09:09 2015
58 Thread 1 is migrating to the core 5 after 128875029 CPU ticks at Sat Mar 14 15:09:09 2015
59 Thread 1 is migrating to the core 3 after 304949136 CPU ticks at Sat Mar 14 15:09:10 2015
60 Thread 3 is migrating to the core 5 after 1711549976 CPU ticks at Sat Mar 14 15:09:10 2015
61 Thread 3 is migrating to the core 0 after 58918164 CPU ticks at Sat Mar 14 15:09:10 2015
62 Thread 4 is migrating to the core 5 after 2211222747 CPU ticks at Sat Mar 14 15:09:10 2015
63 Thread 4 is migrating to the core 2 after 232344623 CPU ticks at Sat Mar 14 15:09:10 2015
64 Thread 5 is migrating to the core 5 after 12459632962 CPU ticks at Sat Mar 14 15:09:10 2015
```

Migrace vláken

- migrování vláken lze zabránit pomocí nastavení systemové proměnné
  - `export OMP_PROC_BIND=TRUE`
  - tento příkaz zadáme na příkazovou řádku před spuštěním našeho programu
- s jejím použitím nám dá stejný program následující výpis

```
1  Press Ctrl-C to quit.
2  Thread 0 is migrating to the core 0 after 292506 CPU ticks at ...
3  Thread 2 is migrating to the core 2 after 816622 CPU ticks at ...
4  Thread 3 is migrating to the core 3 after 1616438 CPU ticks at ...
5  Thread 4 is migrating to the core 4 after 1971582 CPU ticks at ...
6  Thread 5 is migrating to the core 5 after 2743793 CPU ticks at ...
7  Thread 1 is migrating to the core 1 after 3018228 CPU ticks at ...
```