Ε-85: Ειδικά Θέματα Λογισμικού
Προγραμματισμός Συστημάτων Υψηλών Επιδόσεων

Χειμερινό Εξάμηνο 2009-10

«Υπολογιστικά Συστήματα Υψηλών Επιδόσεων και Εφαρμογές»

Παναγιώτης Χατζηδούκας
(Π.Δ. 407/80)

Περίληψη

- Υπερυπολογιστές
- Πολυεπεξεργαστικά συστήματα και νέες τεχνολογίες
- Μοντέλα προγραμματισμού
- Παραδείγματα εφαρμογών

Υπερυπολογιστές

- Οι υπερυπολογιστές (supercomputers) είναι οι πιο ισχυροί υπολογιστές
- Χρησιμοποιούνται για προβλήματα που απαιτούν περίπλοκους και χρονοβόρους υπολογισμούς
- Εξαιτίας του μεγέθους και του κόστους τους, οι υπερυπολογιστές είναι σχετικά σπάνιοι
- Οι υπερυπολογιστές χρησιμοποιούνται από πανεπιστήμια, ερευνητικά κέντρα και μεγάλες επιχειρήσεις

Υπολογιστικά προβλήματα

- Μοντελοποίηση πολύπλοκων διεργασιών
  - Πυρηνική σύντηξη
  - Περιβαλλοντολογικές συνθήκες
- Προβλήματα ρευστοδυναμικής
- Χαρτογράφηση DNA
- Πρόβλεψη καιρού
- Εξόρυξη δεδομένων

TOP 500 (Ιούνιος 2007)

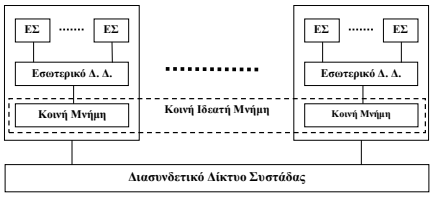
Rank	Site	Computer	Processors	Year	R_{max}	R_{peak}
1	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution IBM	131072	2005	280600	367000
2	Oak Ridge National Laboratory United States	Jaguar - Cray XT4/XT3 Cray Inc.	23016	2006	101700	119350
3	NNSA/Sandia National Laboratories United States	Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core Cray Inc.	26544	2006	101400	127411
4	IBM Thomas J. Watson Research Center United States	BGW - eServer Blue Gene Solution IBM	40960	2005	91290	114688
5	Stony Brook/BNL, New York Center for Computational Sciences United States	New York Blue - eServer Blue Gene Solution IBM	36864	2007	82161	103219
6	DOE/NNSA/LLNL United States	ASC Purple - eServer pSeries p5 575 1.9 GHz IBM	12208	2006	75760	92781
7	Rensselaer Polytechnic Institute, Computational Center for Nanotechnology Innovations United States	eServer Blue Gene Solution IBM	32768	2007	73032	91750
8	NCSA United States	Abe - PowerEdge 1955, 2.33 GHz, Infiniband Dell	9600	2007	62680	89587.2
9	Barcelona Supercomputing Center Spain	MareNostrum - BladeCenter JS21 Cluster, PPC 970, 2.3 GHz, Myrinet IBM	10240	2006	62630	94208
10	Leibniz Rechenzentrum Germany	HLRB-II - Altix 4700 1.6 GHz SGI	9728	2007	56520	62259.2

TOP 500 (Ιούνιος 2009)

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2008 IBM	129600	1105.00	1456.70	2483.47
2	Oak Ridge National Laboratory United States	Jaguar - Cray XT5 QC 2.3 GHz / 2008 Cray Inc.	150152	1059.00	1381.40	6950.60
3	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	825.50	1002.70	2268.00
4	NASA/Ames Research Center/NAS United States	Pleiades - SGI Altix ICE 8200EX, Xeon QC 3.0/2.66 GHz / 2008 SGI	51200	487.01	608.83	2090.00
5	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	478.20	596.38	2329.60
6	National Institute for Computational Sciences/University of Tennessee United States	Kraken XT5 - Cray XT5 QC 2.3 GHz / 2008 Cray Inc.	66000	463.30	607.20	
7	Argonne National Laboratory United States	Blue Gene/P Solution / 2007 IBM	163840	458.61	557.06	1260.00
8	Texas Advanced Computing Center/Univ. of Texas United States	Ranger - SunBlade x6420, Opteron QC 2.3 Ghz, Infiniband / 2008 Sun Microsystems	62976	433.20	579.38	2000.00
9	DOE/NNSA/LLNL United States	Dawn - Blue Gene/P Solution / 2009 IBM	147456	415.70	501.35	1134.00
10	Forschungszentrum Juelich (FZJ) Germany	JUROPA - Sun Constellation, NovaScale R422-E2, Intel Xeon X5570, 2.93 GHz, Sun M9/Mellanox QDR Infiniband/Partec Parastation / 2009 Bull SA	26304	274.80	308.28	1549.00

BlueGene/L (#1 το 2007)

Συστάδες Πολυεπεξεργαστικών Συστημάτων



- Υβριδικό Μοντέλο Μνήμης
  - Κατανεμημένη μνήμη μεταξύ των κόμβων
  - Κοινή Μνήμη μεταξύ των επεξεργαστών ενός κόμβου
- Κοινή Ιδεατή Μνήμη: ενοποίηση των μνημών με κατάλληλο λογισμικό

Πολυεπεξεργαστικά Συστήματα

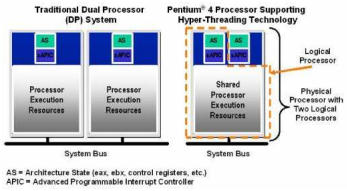
- Πολυεπεξεργασία είναι η χρήση δύο ή περισσότερων επεξεργαστών (CPUs) σε ένα απλό υπολογιστικό σύστημα
- Όλοι οι επεξεργαστές έχουν ίση πρόσβαση στην μνήμη του συστήματος
- Δύο η περισσότερα προγράμματα, π.χ. διεργασίες, εκτελούνται ταυτόχρονα
- Μέχρι πριν λίγα χρόνια, τα συγκεκριμένα συστήματα είχαν αρκετά υψηλό κόστος

Πολυεπεξεργαστικά Συστήματα

- Οι υπολογιστικές ανάγκες των χρηστών συνεχώς αυξάνονται
- Ανάγκη για παράλληλη επεξεργασία χωρίς επιπλέον κόστος
  - Σε επίπεδο υλικού (π.χ. ειδικές μητρικές πλακέτες)
  - Σε επίπεδο λογισμικού (π.χ. άδειες λειτουργικού συστήματος)
- Λύσεις:
  - Επεξεργαστές με τεχνολογία Hyper-threading (SMT)
  - Επεξεργαστές με τεχνολογία Multi-Core

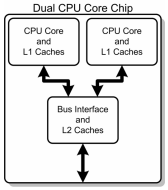
Τεχνολογία Hyper-Threading

- Το σύστημα διαθέτει έναν φυσικό επεξεργαστή με δύο λογικούς επεξεργαστές
- Διαμοίραση των λειτουργικών μονάδων εκτέλεσης του επεξεργαστή

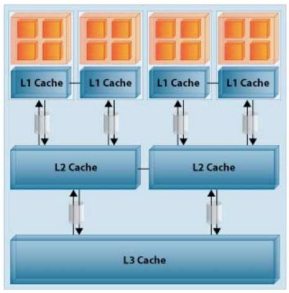


Τεχνολογία Multi-Core

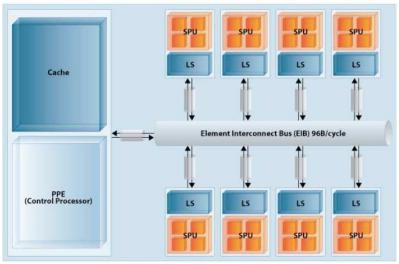
- Στην περίπτωση αυτή έχουμε στο ίδιο chip, δύο ή περισσότερους επεξεργαστικούς πυρήνες
- Απόδοση παρόμοια με αυτή των παραδοσιακών πολυεπεξεργαστικών συστημάτων



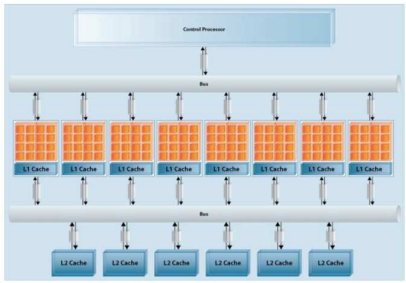
Γενική Πολυπύρηνη Αρχιτεκτονική



Αρχιτεκτονική Cell Broadband Engine



Αρχιτεκτονική NVIDIA GPU G80



Εμπορικά Πολυπύρηνα Συστήματα

Name	Clovertwn	Opteron	Cell	Niagara 2
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8	1*8 = 8
Architecture	4-/3-issue, SSE3, OOO, caches		2-VLIW, SIMD,RAM	1-issue, MT,cache
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz	1.4 GHz
Peak MemBW	21 GB/s	21 GB/s	26 GB/s	41 GB/s
Peak GFLOPS	74.6 GF	17.6 GF	14.6 GF	11.2 GF

Θέματα Λογισμικού

- Απαραίτητη η υποστήριξη από το λειτουργικό σύστημα
- Πολυπρογραμματισμός: ταυτόχρονη εκτέλεση πολλών ανεξάρτητων προγραμμάτων
- Βασικές τεχνικές προγραμματισμού για την εκμετάλλευση των πολλαπλών επεξεργαστών - πυρήνων
  - Διεργασίες (μεγάλο κόστος, δυσκολία προγραμματισμού)
  - Νήματα (μικρό κόστος, ευκολία προγραμματισμού)
  - OpenMP (έμμεση χρήση νημάτων, ελάχιστη προσπάθεια)
  - MPI (ρουτίνες ρητής μετακίνησης δεδομένων μεταξύ διεργασιών, δυσκολία προγραμματισμού)

Υπολογισμός του π – ακολουθιακή έκδοση

```
static long num_steps;
double step;
int main ()
{     int i;
      double x, pi, sum = 0.0;
      num_steps = 100000;

      step = 1.0/(double) num_steps;

      for (i=1;i<= num_steps; i++){
            x = (i-0.5)*step;
            sum = sum + 4.0/(1.0+x*x);
      }
      pi = step * sum;

      return 0;
}
```

Υπολογισμός του π – πολυνηματική έκδοση

```
#include <pthread.h>
#define NUM_THREADS 2
pthread_t thread[NUM_THREADS];
pthread_mutex_t UpdateMutex;
static long num_steps;
double step;
double global_sum = 0.0;

void *Pi (void *arg)
{
  int i, start;
  double x, sum = 0.0;

  start = *(int *) arg;
  step = 1.0/(double) num_steps;

  for (i=start;i<= num_steps;
      i=i+NUM_THREADS){
       x = (i-0.5)*step;
      sum = sum + 4.0/(1.0+x*x);
  }
  pthread_mutex_lock (&UpdateMutex);
  global_sum += sum;
  pthread_mutex_unlock(&UpdateMutex);

  return 0;
}
```

```
int main ()
{
  double pi; int i;
  num_steps = 100000;

  int Arg[NUM_THREADS];

  for(i=0; i<NUM_THREADS; i++)
     threadArg[i] = i+1;

  pthread_mutex_init (&UpdateMutex, NULL);

  for (i=0; i<NUM_THREADS; i++)
     pthread_create(&thread[I], NULL, Pi,&Arg[i]);


  for (i=0; i<NUM_THREADS; i++)
     pthread_join(thread[i], NULL);

  pi = global_sum * step;

  return 0;
}
```


Υπολογισμός του π – ακολουθιακή έκδοση

```
static long num_steps;
double step;
int main ()
{     int i;
      double x, pi, sum = 0.0;
      num_steps = 100000;

      step = 1.0/(double) num_steps;

      for (i=1;i<= num_steps; i++){
            x = (i-0.5)*step;
            sum = sum + 4.0/(1.0+x*x);
      }
      pi = step * sum;

      return 0;
}
```


Υπολογισμός του π – με χρήση του OpenMP

```
#include <omp.h>
static long num_steps = 100000;
double step;
#define NUM_THREADS 2
int main ()
{     int i;
      double x, pi, sum = 0.0;

      step = 1.0/(double) num_steps;
      omp_set_num_threads(NUM_THREADS);
#pragma omp parallel for reduction(+:sum) private(x)
      for (i=1;i<= num_steps; i++){
            x = (i-0.5)*step;
            sum = sum + 4.0/(1.0+x*x);
      }
      pi = step * sum;

     return 0;
}
```


Υπολογισμός του π – ακολουθιακή έκδοση

```
static long num_steps;
double step;
int main ()
{     int i;
      double x, pi, sum = 0.0;
      num_steps = 100000;

      step = 1.0/(double) num_steps;

      for (i=1;i<= num_steps; i++){
            x = (i-0.5)*step;
            sum = sum + 4.0/(1.0+x*x);
      }
      pi = step * sum;

      return 0;
}
```


Υπολογισμός του π – με χρήση του MPI

```
#include <mpi.h>
static long num_steps;
double step;
int main(int argc,char *argv[]) {
   int i, me, nproc;
   double x, pi, sum=0.0, local_pi;
   MPI_Init(&argc, &argv);
   MPI_Comm_size(
          MPI_COMM_WORLD,
          &numprocs);
    MPI_Comm_rank(
          MPI_COMM_WORLD,
          &myid);

   <core execution code>

   MPI_Finalize();
   return 0;
}
```

```
if (myid == 0) num_steps = 100000;
MPI_Bcast(&numsteps, 1, MPI_INT, 0,
          MPI_COMM_WORLD);

step = 1.0/(double) num_steps;
for (i=1;i<= num_steps; i+=numprocs){
     x = (i-0.5)*step;
     sum = sum + 4.0/(1.0+x*x);
}
local_pi = step * sum;

MPI_Reduce(&local_pi, &pi, 1,
          MPI_DOUBLE, MPI_SUM, 0,
          MPI_COMM_WORLD);
```


OpenMP

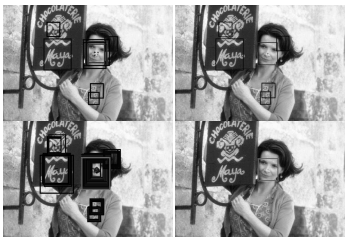
- Μοντέλο προγραμματισμού για παράλληλα συστήματα
- Οδηγίες προς τον μεταγλωττιστή ώστε να παράγει κώδικα με νήματα
- Όλες οι μεγάλες εταιρίες το υποστηρίζουν ή/και αποφασίζουν για την εξέλιξή του
  - Intel, SUN, IBM, …
  - Microsoft (Visual Studio 2005)
  - GNU GCC 4.2
- Ερευνητικοί compilers
  - Omni (Ιαπωνία), NANOS (Ισπανία), OpenUH (ΗΠΑ), OMPi (Ελλάδα)

Παραδείγματα εφαρμογών

- Σύστημα εντοπισμού προσώπου σε εικόνες (face detection)
- Αλγόριθμος ομαδοποίησης δεδομένων (data clustering)

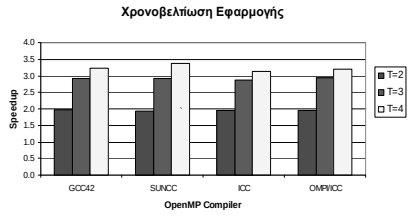
Εντοπισμός προσώπου

- Στόχος η δυνατότητα επεξεργασίας εικόνων σε πραγματικό χρόνο
- Σύστημα εντοπισμού προσώπου: Delakis & Garcia



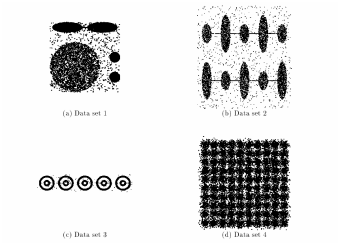
Εντοπισμός προσώπου

- Χρονοβελτίωση σε σύστημα με 4 επεξεργαστές = 3.5
- Επίτευξη: εκτέλεση του αλγορίθμου εντοπισμού προσώπου σε πραγματικό χρόνο σε ένα οικονομικό υπολογιστικό σύστημα.



Ομαδοποίηση δεδομένων

- Διαχώριση παρόμοιων δεδομένων σε συστάδες
- Παραλληλοποίηση ενός αλγορίθμου (CURE) με χρήση του μοντέλου OpenMP



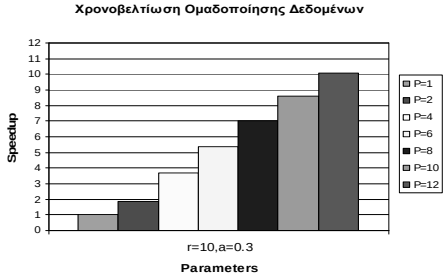
Ομαδοποίηση δεδομένων

- Πολύ μεγάλος αλλά προβλέψιμος χρόνος εκτέλεσης
- Χρονοβελτίωση στους 4 επεξεργαστές: ~3.5x

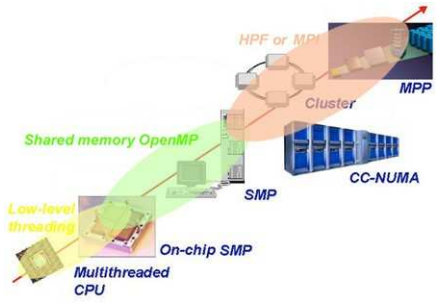
Εγγραφές	10K	100K	1M
Χρόνος (s)	58.5	5885	591472 *
		~ $x10^2$	~ $x10^4$

- (*) Χρόνος εκτέλεσης από 7 σε 2 ημέρες

Ομαδοποίηση δεδομένων



Μοντέλα Προγραμματισμού (σήμερα)



Μοντέλα Προγραμματισμού (μέλλον)