

Cluster computing kurser

Indhold

Cluster computing	2
Moore's lov	3
Moore's lov versus grafikkort	3
Cluster η sammenligning	4
Teknologier	5
Beowulf cluster	5
CUDA programmering	6
Kursusforløb i clusterprogrammering	6
Én-dags introduktionskursus til parallelprogrammering . .	6
Tre-dages introduktionskursus til parallelprogrammering .	7
Dag 1: Introduktion til parallel- og clusterprogramme- ring	8
Dag 2: Beowulf clusters	9
Dag 3: nVidia GP-GPU's	9



ADRESSE: MERGEIT APS
KONGSVANG ALLÉ 37
8000 ÅRHUS C

TELEFON: +45 8820 2078
CVR: 3056 6602

MAIL: INFO@MERGEIT.DK
WEB: WWW.MERGEIT.DK



Cluster computing

Brug af parallelle computersystemer kan gøre tunge udregninger hurtigere og meget mere effektivt.

Traditionelt set består et clustersystem af to eller flere ens maskiner, der er koblet sammen, f.eks. ved et TCP/IP netværk, som således samlet fungerer som én stor computer. Paralleliteten i systemet gør, at udregningshastigheden kan skaleres linært op med antallet af undermaskiner i clusteren.

Brugen af parallelle systemer har på det seneste vundet udbredelse i 'dual-' og 'multi-cores' CPU eller ved brug af dedikerede multi-computere, der på et konceptuelt plan minder om en traditionel cluster, dog med en væsentlig forbedret kommunikationsbåndbredde.

Udfordringerne i brug af clusters eller andre parallelle systemer består hovedsageligt i at programmere disse systemer, således at delementerne, hukommelse, CPU, og kommunikationskanalerne bruges optimalt. Dette er en ikke-triviell opgave, der kræver ekstrem meget viden og erfaring af udvikleren for at opgaverne kan løses succesfuldt.



En Beowulf cluster computer: PPPL Scientific Computing Cluster
(ikke at forveksle med HAL9000)

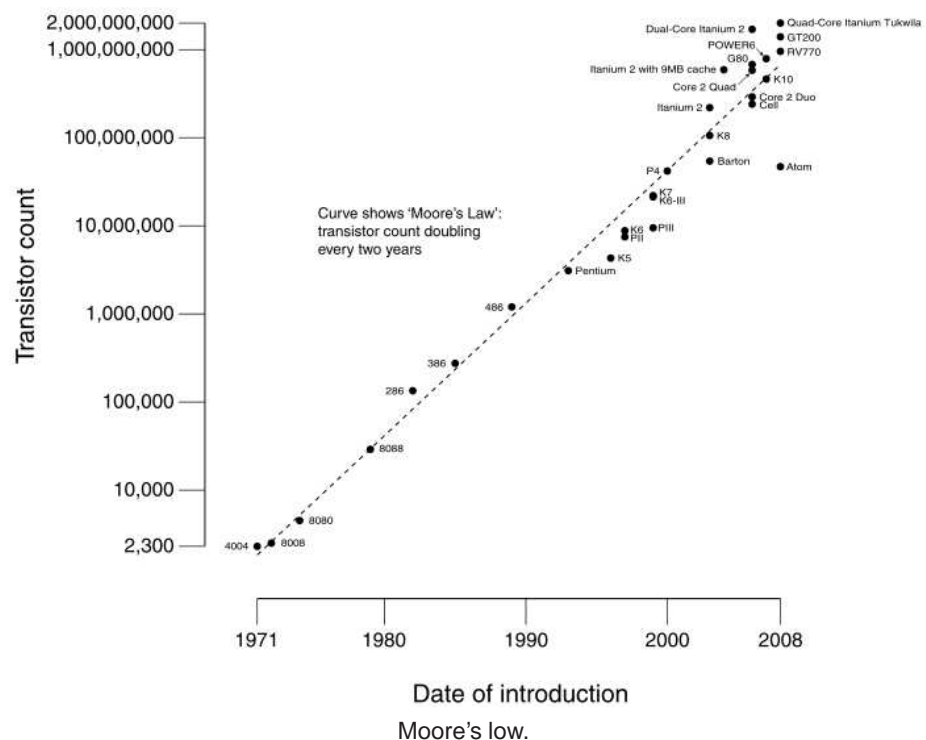
[<http://beowulf.pppl.gov>]



Moore's lov Et typisk mål for en supercomputer eller clustersystem er antal 'floating points' operationer per sekund, FLOPS. Dette mål benyttes primært i forbindelse med regneopgaver, der udføres med flydetal (floats) f.eks. Fourier transformationer. (Alternativt kan man benytte en-million heltals operationer per sekund, MIPS.)

Selvom Moore's lov har holdt et par dekader nu, kan den eksponentielle vækst være tæt ved at bryde sammen. Anderledes er det med systemer, der er opbygget af ens delkomponenter: Den parallelle natur gør, at systemet kan skaleres op ved simpel duplikation af de enkelte komponenter.

CPU Transistor Counts 1971-2008 & Moore's Law

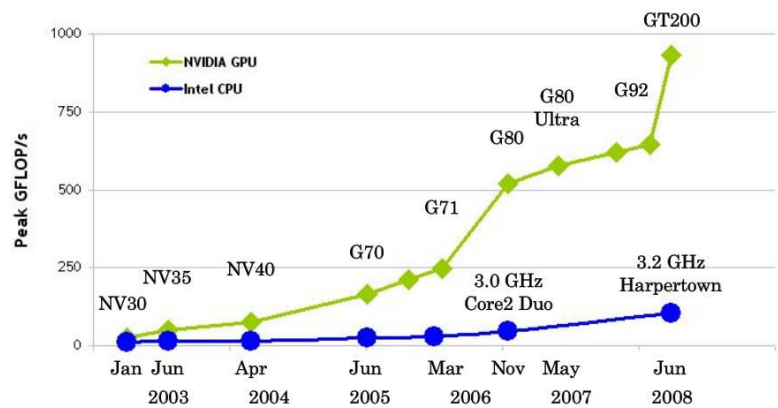


[http://en.wikipedia.org/wiki/File:Transistor_Count_and_Moore%27s_Law_-_2008.svg]

Moore's lov versus grafikkort Et nyt og interessant skud på stammen af parallelsystemer er anvendelsen af grafikkort. På sådan et kort sidder der en hel række dedikerede processorer ('Graphical Programmable Unit, GPU), der oprindeligt blev brugt til hurtig rendering af 3D grafik. Man kan dog programmere disse processorer til mere generelle formål og



en række firmaer tilbyder gratis software, så disse 'General purpose graphical programmable units' (GP-GPU) kan programmeres.



CPU'erne (blå kurve) vs GP-GPU'erne (grøn kurve).

[NVIDIA CUDA, Programming Guide, Version 2.1, 12/8/2008]

Cluster η sammenligning De parallelle systemer skalerer i teorien linært med antallet af processorer, men i praksis er der en række forhindringer, der kan gøre systemerne mindre effektive; der vil være en *overhead* på kommunikationen og samarbejdet processerne imellem, således at man ikke kan regne med 100% effektivitet.

Udregner man forholdet mellem pris og performance

$$\eta = \frac{\text{pris}}{\text{performance}}$$

får man et interessant indblik i forskellige systemers ydeevner, sammenholdt med pris. Det forholder sig således, at de tidligere dedikerede cluster systemer, baseret på f.eks. Itanium processor, performer ret dårligt når man ser på η . Her er nye systemer baseret på kommercielt 'off-the-shelf' hardware (COTS) langt billigere, og i den helt ekstreme ende finder man hardware, baseret på spilkonsoler og grafik kort.



Cluster system	~GFLOPS	~pris	η
Hjemmebygget Beowulf	32	3300 kr	103 kr/GFLOPS
PS3 SONY:	205	3000 kr	15 kr/GFLOPS
nVidia 8400 GS	20	300 kr	15 kr/GFLOPS
nVidia 8800 GTX	500	1954 kr	3.9 kr/GFLOPS
Intel Itanium (priser fra 2002)	3.2	54000 kr	16900 kr/GFLOPS
Intel Itanium 2	6.7	24000 kr	3600 kr/GFLOPS

Udviklingen i hardware, drevet af spilindustrien, er således et fremragende alternativ, både pga. den ekstremt høje performance, men også fordi at konkurrencen og udviklingen har drevet prisen helt i bund.

Der ligger dog en række problemer ved at bruge sådanne dedikerede systemer. Programmeringen af hardwaren kan være vanskelig og har en meget stejl indlæringskurve.

Teknologier

Beowulf cluster



En Beowulf cluster er en teknologi, der udelukkende baserer sig på åbne software systemer og billig PC hardware. Definitionen af en Beowulf lyder noget a lá en cluster baseret på et Open Source operativsystem, udelukkende opbygget med Open Source software komponenter, der kører på billig COTS hardware.

Typisk baseres interkommunikation i en Beowulf sig på MPI over et TCP/IP net, og en Beowulf vil typisk være baseret på rå computere uden skærm, tastatur eller harddisk, for at gøre systemet billigt og robust. En master computer kan kontrollere de enkelte subnodes, der normalt opbygges af fuldstændig ens stykker hardware.

En cluster teknologi, baseret på Beowulf ideene, er således relativt nem at opbygge, vedligeholde og udbygge. Den kan levere en formidabel performance pga. valget af COTS hardware og FOSS software.



CUDA programmering De nyeste grafikkort fra nVidia kan programmeres direkte i C via nVidia's CUDA udviklingsmiljø (Compute Unified Device Architecture).

De seneste versioner af nVidia grafikkort leverer en hidtil uset performance, der potentielt set kan give os supercomputer kræfter i TFLOPS klassen, vel at mærke til en pris der ligger flere magnituder under traditionelle supercomputer systemer.

En *hurdle*, før man når supercomputer Nirvana, er dog at man kan formulere sit problem, så det passer til GPU'en. Dette er en ikke-trivielt opgave, idet programmeringsmiljøet i CUDA, trods den umiddelbare lighed med C, kræver en radikal ændring af normal programmeringstankegang.

Kursusforløb i clusterprogrammering

Én-dags introduktionskursus til parallelprogrammering Et 'crash-course' i parallelprogrammering; teknologier, metoder og programmering.

Der sigtes på at give kursisten en generel introduktion til forskellige metoder og værktøjer, der typisk anvendes inden for parallelprogrammering.

Herefter vil vi gå i detaljer med to specifikke parallelprogrammeringsteknologier: et 'message passing' system, baseret på MPI og Beowulf clusters, og et 'shared memory' system, baseret på nVidia's grafikkort.

Kurset vil være tilegnet både begyndere og let øvede i teknologierne, med en sværhedsgrad, der løbende stiger igennem kurset.

Der vil primært blive vist kode i C og C++, og kurset kræver ingen forberedelse.

- Introduktion til parallelprogrammering.
 - Multithreading.
 - Brug af globale variable.
 - Multicores CPU's og multithreading.
 - Superthreading og hyperthreading.
- Cluster teknologi og metoder.



- High-performance-, cluster-, distribueret- og gridcomputing.
- Software metoder: message passign vs. shared memory.
- Modeller: implicit/eksplicit parallelitet, concurrency.
- Task/Data parallisme.
- Flynn's taksonomi: SISD, SIMD, MISD, MIMD
- Amdahl's lov, Gustafson's lov

- Middagspause.

- Beowulf clusters
 - Beowulf COTS clusters forklaret.
 - MPI message passing
 - Parallel programmering i C++/MPI.
 - Hukommelsesmodel og distribuerede algoritmer
 - Eksempel 1: Parallel Fourier transformering via FFTW.
 - Eksempel 2: Kosmologisk simulering via Gadget2.
 - Eksempel 3: Afstand fra alle-punkter-til-alle-andre-punkter (to-punks-korrelationsfunktion).

- nVidia GP-GPU's
 - Introduktion til CUDA udviklingsmiljøet.
 - Memorymodel i CUDA, lokal, global, shared.
 - Memorytilgang og læse/skrive hastighed.
 - Kommunikation med GPU'en.
 - CUDA programmerings-primitiver.
 - Parallelisering af algoritmer
 - Data tilgang og optimering.
 - Debug af GPU programmer.

Tre-dages introduktionskursus til parallelprogrammering

Kurset giver et dybere indblik i parallelprogrammering, og vi vil komme bredt omkring forskellige parallelprogrammeringsteknologier, metoder og programmeringsteknikker.

Kurset består af en generel introduktion til cluster og parallelprogrammering, og vil herefter gå i detaljer med to specifikke parallelprogrammeringsteknologier: et 'message passing' system, baseret på MPI og Beowulf clusters, og et 'shared memory' system, baseret på nVidia's grafikort.

Kurset er rettet mod let- til meget-øvede programmører, der til dagligt benytter sig af C eller C++. Der vil inden kurset blive udleveret



materiale, som giver kursisten en kort generel introduktion til de begreber, der vil blive diskuteret på kurset.

Der vil på kurset blive udlevet opgaver til 'hands-on' øvelser, der kan løses af kursisterne som par-programmering (dvs. to-og-to programmører sammen).

Der vil blive stillet computere til rådighed i kursusforløbet, med Linux, C++, MPI, og CUDA miljøerne preinstalleret, men kendskab til Linux er dog ikke en nødvendig forudsætning.

Dag 1: Introduktion til parallel- og clusterprogrammering



- Multithreadprogrammering: Med udgangspunkt i C++ og Boost::Threads gives en introduktion til programmering i multithread miljøer.
 - Process-begreber: fibers, threads og processer.
 - OS kernebegreber: timeslicing, (preemptive) multitask kerne, realtidskerner og latency.
 - Multithreading. primitiver 1: mutex/semaphore.
 - Multithreading. primitiver 2: fork/join,yield, sleep.
 - Thread-safe kode.
 - Brug af globale variable.
 - Multicores CPU's og multithreading.
 - Superthreading og hyperthreading.
- Cluster teknologi og metoder.
 - Introduktion til High-performance-, cluster-, distribueret- og gridcomputing.
 - Distributed og cluster computing i detaljer.
 - Software metoder: message passign vs. shared memory.
 - Task/Data parallisme.
 - Modeller: implicit/eksplicit parallelitet, concurrency.
 - Flynn's taksonomi: SISD, SIMD, MISD, MIMD
 - Grid computing vs. clusters.
 - Amdahl's lov, Gustafson's lov.
 - Hardware: superscalar, multicores, vector processor.



Dag 2: Beowulf clusters

- Introduktion til Beowulfs.
 - Cluster setup.
 - MPI message passing.
 - MPI setup.
 - MPI primitiver, barrier, synchronization, scatter/gather.
 - C++ marshalling interface via MPI.
 - Hands-on øvelse: MPI og marshalling 1.
 - Parallel programmering i C++/MPI.
 - Hukommelsesmodel og distribuerede algoritmer
 - Hands-on øvelse: MPI og marshalling 2.
- Middagspause.
 - Eksempel 1: Parallel Fourier transformering via FFTW.
 - Eksempel 2: Kosmologisk simulering via Gadget2.
 - Hands-on øvelse: et lille MPI/Beowulf system 1.
 - Eksempel 3: Afstand fra alle-punkter-til-alle-andre-punkter (to-punks-korrelationsfunktion).
 - Hands-on øvelse: et lille MPI/Beowulf system 2.

Dag 3: nVidia GP-GPU's

- Introduktion til nVidia GP-GPU programmering.
 - Introduktion til CUDA udviklingsmiljøet.
 - Memorymodel i CUDA, lokal, global, shared.
 - Memorytilgang og læse/skrive hastighed.
 - Kommunikation med GPU'en.
 - Hands-on øvelse: et lille GPU program.
- Middagspause.
 - CUDA programmeringsprimitiver.
 - Parallelisering af algoritmer
 - Hands-on øvelse: udvidelse af det lille GPU program.
 - Data tilgang og optimering.
 - Debug af GPU programmer.
 - Hands-on øvelse: en GPU baseret matriks-multiplikator