DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Taming the multi-core beast

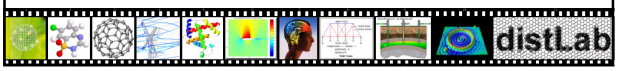
Professor Brian Vinter

Concurrency vs. Parallelism

Designing parallel systems is very hard
- But sometimes necessary
- Designing concurrent systems is surprisingly easy

Concurrent systems can transparently utilize an underlying parallel system

If your program is concurrent, and with sufficient concurrent operations, you don't have to design for a specific number of processors



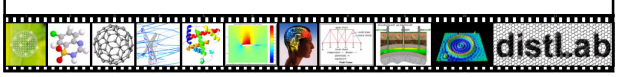
Parallelism

News flash: Moore's law is dead
- Since October 2004

This means that increased performance must come from using more processors

How much of a Pentium 4 processor is actual processing power?

Next generation processors are all multi-core

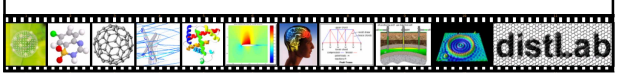


Concurrency

Concurrency it the natural way of things
Serial algorithms are just the way we are trained…

All (most) new CPUs are hardware threaded
- Why? (hint: how far is a nano-second?)

The basic assumption behind hardware threading is that your application is multithreaded
- Otherwise there is little advantage….



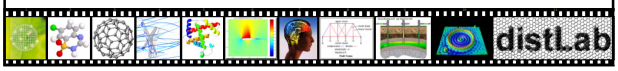
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Natural Concurrency

Example: Network server

If a network server is servicing many clients

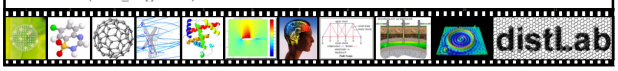
- How to figure out which is active?
- How to prioritize connections?



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Network server: In olden times

```
listen(socket)
for i in clients:
   socket[i]=accept(socket)

for (i=0; i< num_clients; i++)
   FD_SET(sockets[i], &recvfds);

select(clients+1, recvfds, NULL, NULL, NULL)

j=0
for (i=0; i< num_clients; i++)
   if FD_ISSET(recvfds, i){
      add_to_active(i) //ok we will cheat here and use blackbox code
      j++
   }

my_qsort(active_set)//And some more blackbox code her

for (i=0; i<j; i++)
service(active_set[i].socket)
```

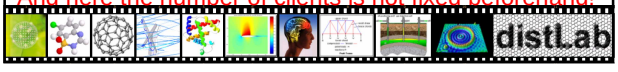


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Network server: Now

```
listen(socket,1)
while (1){
   client = accept(socket)
   thread(priority, service, client)
}
```

And here the number of clients is not fixed beforehand!



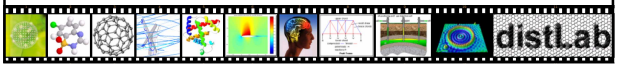
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Java Threading

`<java.sun.com/products/jfc/tsc/articles/threads/threads1.html>`

'If you can get away with it, avoid using threads. Threads can be difficult to use, and they make programs harder to debug."

'Component developers do not have to have an in-depth understanding of threads programming: toolkits in which all components must fully support multithreaded access, can be difficult to extend, particularly for developers who are not expert at threads programming."

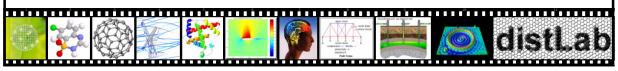


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Java Threading

<java.sun.com/products/jfc/tsc/articles/threads/threads1.html>

''It is our basic belief that extreme caution is warranted when designing and building multi-threaded applications … use of threads can be very deceptive … in almost all cases they make debugging, testing, and maintenance vastly more difficult and sometimes impossible. Neither the training, experience, or actual practices of most programmers, nor the tools we have to help us, are designed to cope with the non-determinism … this is particularly true in Java … we urge you to think twice about using threads in cases where they are not absolutely necessary …''



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Niagra

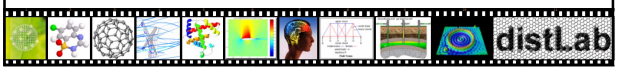
Suns new CPU
- Designed for Java

8 processor cores on a CPU

32 HW threads on a CPU

So while threading should be avoided - according to Sun- you do need at least 32 to make the thing perform

We should aim at millions – just to be sure we are ready for the next generation CPU

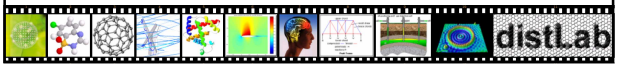


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Communicating Sequential Processes

A mathematical theory for specifying and verifying complex patterns of behavior arising from interactions between concurrent objects.

CSP has a formal, and compositional, semantics that is in line with our informal intuition about the way things work.

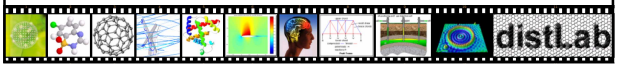


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

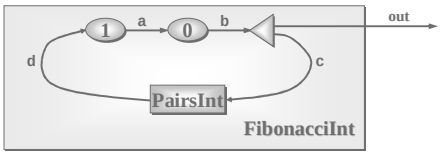
So, what is CSP?

CSP deals with ***processes***, ***networks*** of processes and various forms of ***synchronization*** / ***communication*** between processes.

A network of processes is also a process - so **CSP** naturally accommodates layered network structures (***networks of networks***).



CSP Example




```
FibonacciInt (out) = PrefixInt (1, d, a) ||
                     PrefixInt (0, a, b) ||
                     Delta2Int (b, out, c) ||
                     PairsInt (d, c)
```

0 1 1 2 3 5 8 13 21 34 ...

Verification

A powerful feature in concurrent design is composition

If you design very small functionalities in each process you can verify their correctness

If you combine such small processes in a communicating network you can verify that that network is correct

You then have a new (larger) process you know to be correct and can be used to build larger components again

Motivating Example

Most hardware is build using concurrency designs because:

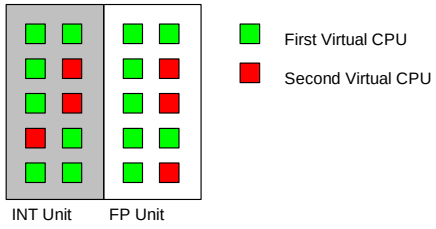
- Parallelism – all the hardware exist at all times
- Concurrency – you cannot determine the order and timing of all events
- Verification – you cannot distribute a patch to existing hardware

The Pentium processor and Windows 95 were siblings

- Both with app 15MLOC

We found 5 errors in the Pentium.....

Hyperthreading



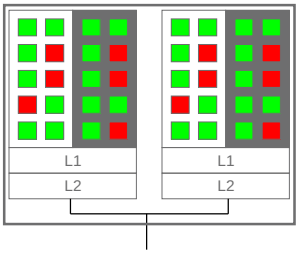
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Multi-core architectures

Multi-cores comes out of necessity
Some are well planned
Some are hacks

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

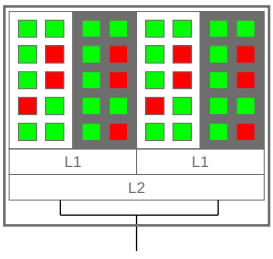
Dual-core (trivial)



distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

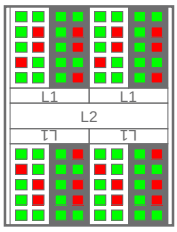
Dual-core



distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Quad-core

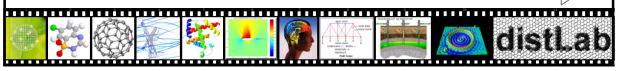


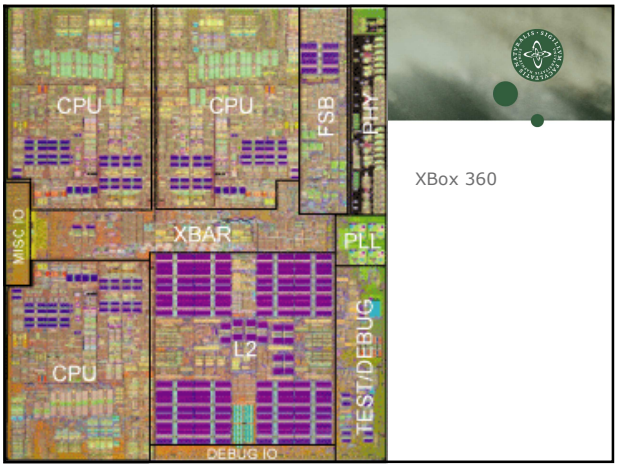
distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Check time!!!

Skip ahead





XBox 360

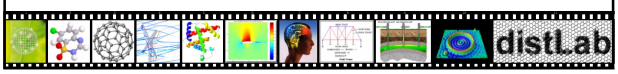
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Components

3 PowerPC cores
- 2 threads pr core
- 3.2 GHz

R500 Graphics Processor
- 500MHz
- 48 Pipelines

256 MB RAM

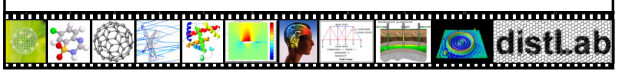


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Xenon

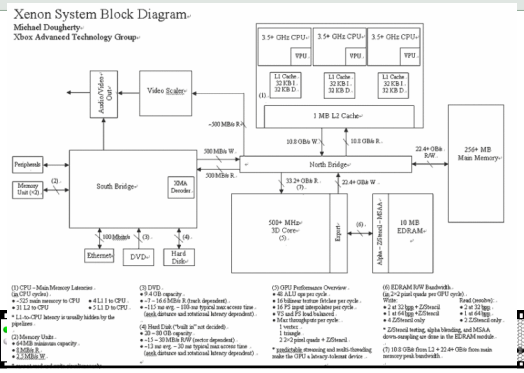
3-Way Symmetric Multi-Processor
- IBM PowerPC Architecture®
- Specialized Function VMX
- 3.2GHz
  - Shared 1 MByte L2
  - Front Side Bus / PHY – 21.6 GB/sec
  - Phase Locked Loops

165 M Transistors
- IBM – 90nm SOI



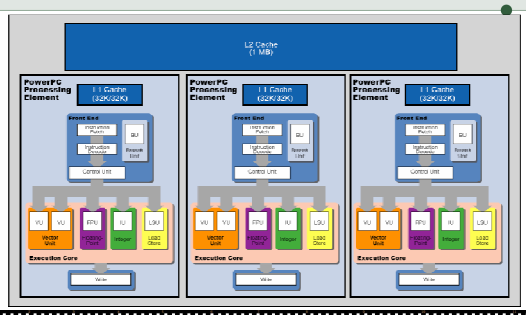
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Layout



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Layout of the Xenon



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

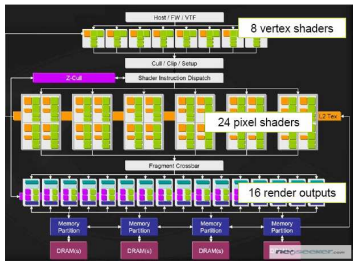
GPU

Custom ATI Graphics Processor

- 10MB DRAM
- 48-way parallel floating point
- Unified shader architecture
- 500 million triangles per sec
- 16 gigasamples/sec
- 48 billion shader operations/sec

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

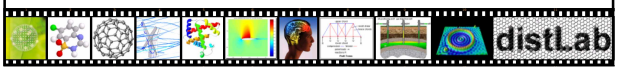
GPU



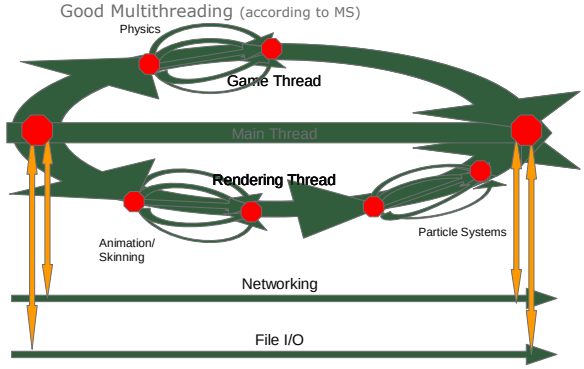
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Memory

256 MB of 700MHz GDDR3 RAM – unified memory architecture
22.4 GB/s interface bus bandwidth
256 GB/s memory bandwith to EDRAM
21.6 GB/s front-side bus



Good Multithreading (according to MS)



Another Paradigm: Cascades

Advantages:
- Synchronization points are few and well-defined

Disadvantages:
- Increases latency

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

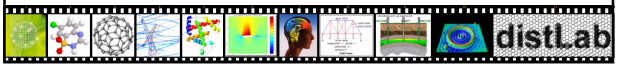
Basic Thread Management

CreateThread()
- Watch *dwStackSize*

SuspendThread(), ResumeThread()
- Probably a bad idea—can lead to deadlocks

TerminateThread() – not available
XSetThreadProcessor()
- proc / 2 = core
- proc % 2 = hw thread

WaitForSingleObject(), CloseHandle()

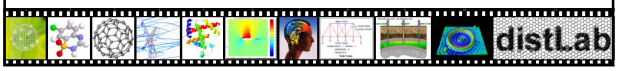


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Fibers

Cooperative software pseudothreads
Do not preempt
ConvertThreadToFiber()
CreateFiber()
SwitchToFiber()

Context switches are 7-9× faster than software threads

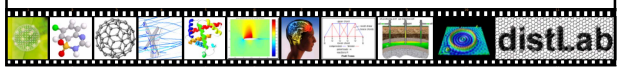


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CSP

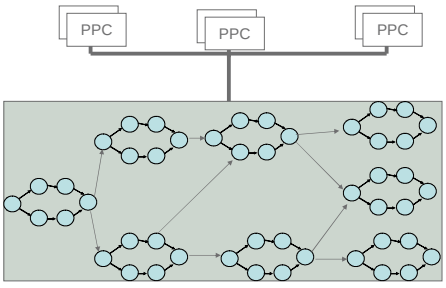
Program your application as a CSP network
- Make sure you have enough processes
  - Enough >> 6

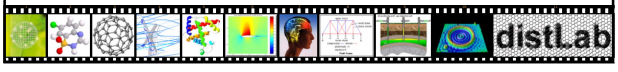
With shared cache we can let all 6 HW threads use the same scheduler
- But we can also let L1 dictate a 2x3 scheduler



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CSP



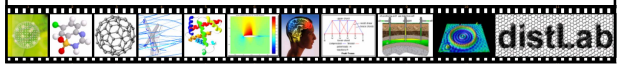


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Advantages of CSP

No consideration of the underlying architecture when determining parallelism
- Porting to other architectures is easy

Dynamic load balancing



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Problems with CSP

No modern tools exists

CPS kernel must be implemented with knowledge of the architecture

- This should be really easy on this architecture though

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Problems with CSP

~~No modern tools exists~~

CPS kernel must be implemented with knowledge of the architecture

- This should be rea… though

News Flash:
CSP.Net

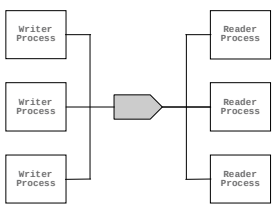
(www.cspdotnet.com)

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CSP.NET

Channel

The CSP.NET channel is a generic rendez-vous channel



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CSP.NET

Example - ReaderWriter

```
public class Writer : Process
{
    private IChannelOut<int> chan;

    public Writer(IChannelOut<int> chan)
    {  this.chan = chan;
    }

    public override void Run()
    {  for (int i = 0; i < 100; i++)
           chan.Write(i);
    }
}
```

CSP.NET

Example - ReaderWriter

```
public class Reader : Process
{
    private IChannelIn<int> chan;

    public Reader(IChannelIn<int> chan)
    {
        this.chan = chan;
    }

    public override void Run()
    {
        for (int i = 0; i < 50; i++)
            Console.WriteLine(chan.Read());
    }
}
```


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CSP.NET

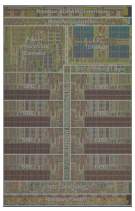
Example - ReaderWriter

```
class Program
{
    static void Main(string[] args)
    {
        CspManager.StackSize = 131072;
        Channel<int> chan = new Channel<int>();
        new Parallel(new Writer(chan),
                     new Reader(chan),
                     new Reader(chan)).Run();
    }
}
```

distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

HPC for Fun and Profit



The CELL processor and the challenges it will present to us..

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Whatever happened with the P4@4GHz?

It should have arrived in October 2004

- The cancellation equaled the death of Moores law
- Which is not really true

What are we going to do for performance?

distLab

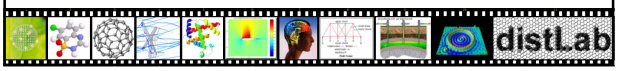
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

The challenges

Memory Wall
- Latency induced bandwidth limitations

Power Wall
- Must improve efficiency and performance equally

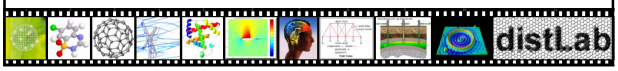
Frequency Wall
- Diminishing returns from deeper pipelines



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Supercomputing

Year	Rank	Manufacturer	Computer	Procs	Rpeak
1993	1	TMC	CM-5	1024	131
2002	500	IBM	Power3	132	198

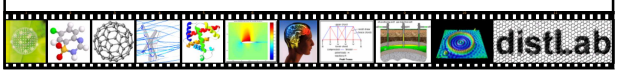


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CELL

Next generation CPU from
- Sony
- Toshiba
- IBM

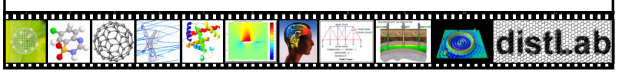
Heterogeneous multi-core CPU



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Basics

Multi-core microprocessor (9 cores)
~250M transistors
~235mm2
Top frequency >5 GHz
9 cores, 10 threads
> 200+ GFlops (SP) @3.2 GHz
> 20+ GFlops (DP) @3.2 GHz
Up to 25.6GB/s memory B/W
Up to 50+ GB/s I/O B/W



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CELL Philosophy

If something is not inherently good
Why keep adding it?

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Addressing the challenges with CELL

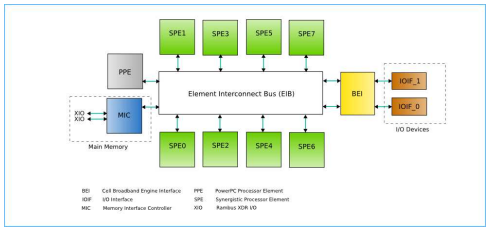
Multi-Core Non-Homogeneous Architecture
- Control Plane vs. Data Plane processors
- Attacks Power Wall

3-level Model of Memory
- Main Memory, Local Store, Registers
- Attacks Memory Wall

Large Register File & SW Controlled Branching
- Allows deeper pipelines
- Attacks Frequency Wall

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CELL Overview



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

PPE

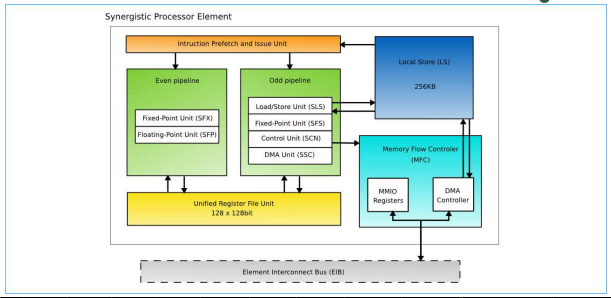
64 bit Power CPU
128 bit VMX
- If you want this thing to do graphics

32 KB + 32 KB Level 1 cache
- Instruction and data

512 KB Level 2 cache
2 hardware threads
In order execution so no need for `lwsync`

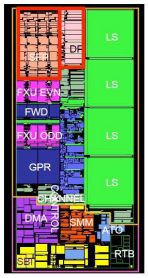
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

SPE



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

SPE



RISC like organization
- 32 bit fixed instructions
- Clean design – unified Register file

User-mode architecture
- No translation/protection within SPU
- DMA is full Power Arch protect/x-late

VMX-like SIMD dataflow
- Broad set of operations (8 / 16 / 32 Byte)
- Graphics SP-Float
- IEEE DP-Float

Unified register file
- 128 entry x 128 bit

256KB Local Store
- Combined I & D
- 16B/cycle L/S bandwidth
- 128B/cycle DMA bandwidth

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

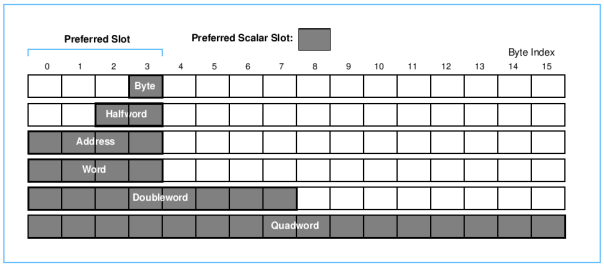
The SPE Register file

128 general purpose registers
- 2KB!!!

May be used in any size

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

The SPE Register file



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

SPE is a SCB

Instructions	Execution Pipe	Latency
Word arithmetics	Even	2
Word shift and rotate	Even	4
SP Multiply add	Even	6
Int Multiply add	Even	7
Byte ops	Even	4
Quadword shift,etc	Even	4
Load/Store	Odd	6
Channel access	Odd	6
Branch	Odd	3

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Local store

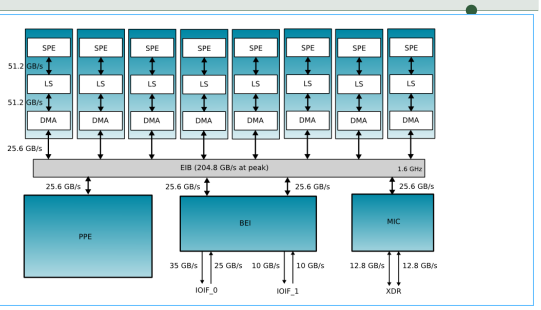
Default Size is 256KB / SPE

Using the Local Store is going to be all important for performance

- Load from LS in 6 cycles
- Enqueue DMA transfer in 20 cycles
- 16B/cycle L/S bandwidth
- 128B/cycle DMA bandwidth

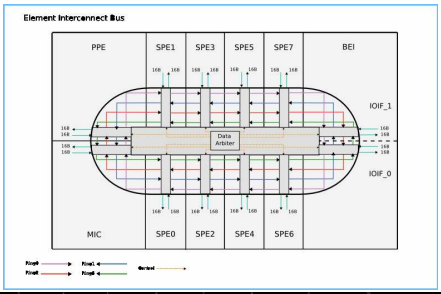
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Data capacity



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

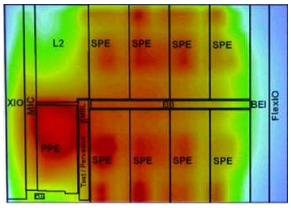
The Network



Power

Power consumption is in the 80W range

- But it seems to be used correctly



Programming the beast

For optimal performance we need to keep 10 threads running

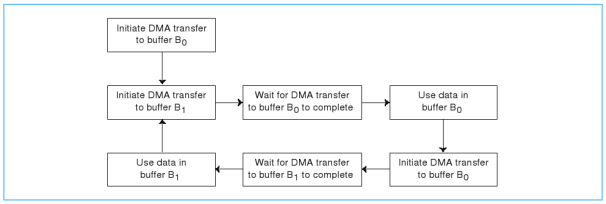
- 2 PPC threads
- 8 SPE threads

For these threads to run wee need to keep filling the Local Stores

- Thus another 8 DMA engines

This means we need to keep 18 functional units active at any time!!!

DMA transfers at all times…



The responsibility returns to the programmer

Now we have to fetch the memory in advance

- No more relying on caches for performance
- No more having to write your code to match the cache behavior

Now we have to do branch prediction

- You have to hint the CPU if a branch is taken or not

The SPE has two special pipelines

- One for load, store and branch
- One for arithmetic operations

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

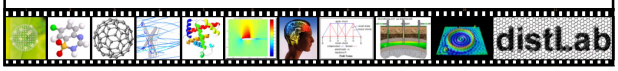
Programming the CELL

Slice the processor
- Use each SPE for something special

Pipeline the processor
- Combine the SPEs for execution

Bag of tasks

CSP

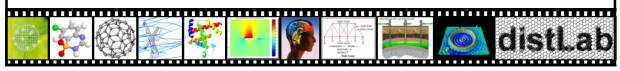


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Sliced CELL

Take the SPE processors and use them as special purpose processors

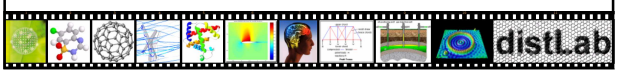
Each processor is dedicated to each processor



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Sliced CELL

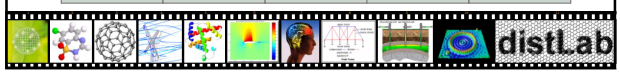
SPE SPE SPE SPE SPE



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Sliced CELL

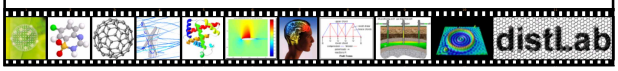
AI	Physics	Action	Graphics	Sound
SPE	SPE	SPE	SPE	SPE



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Advantages of slicing

Really really easy
Easy to develop and test in components

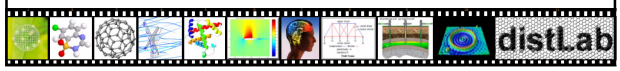


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Problems with slicing

Will you have exactly 7 dedicated tasks?
- More than 7?
- Less than 7?

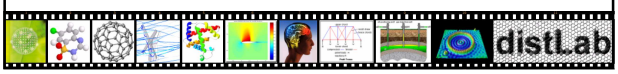
Will each task require equal amount of processing power?
Will all tasks be able to run at the same time?
What is the communication between tasks?



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

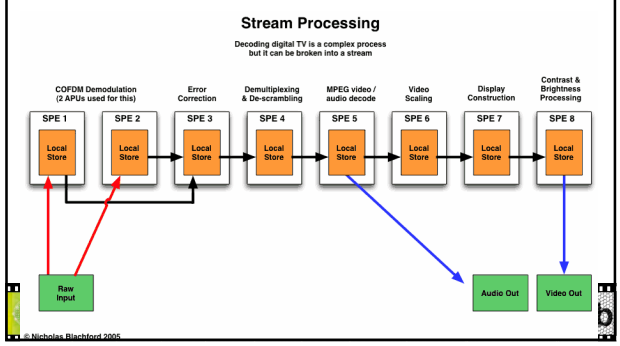
Pipelined CELL

Divide your application into a software pipeline
Complex operations are split over multible SPEs



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Pipelined CELL

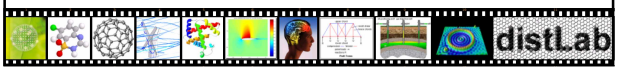


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Advantages of pipelining

Allow high 'operations per second' for inherently sequential code

Only nearest neighbor communication



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

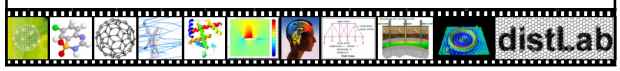
Problems with pipelining

Splitting operations in a pipeline is often hard and sometimes impossible

Making sure that all steps are equally hard is impossible

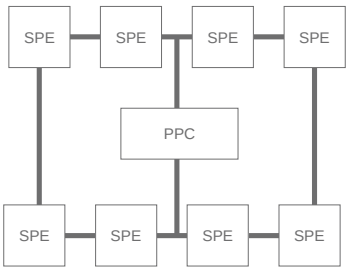
Communication can be very high

- Because we are exchanging intermediate data



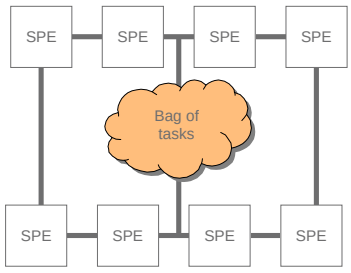
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Bag of Tasks



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Bag of Tasks



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

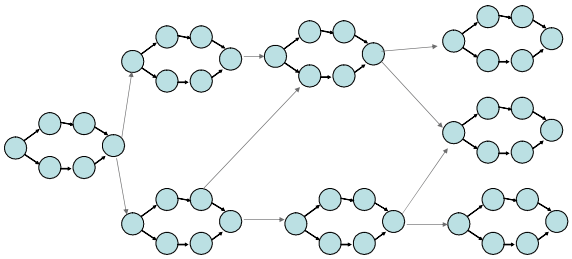
CSP the CELL

Program your application as a CSP network
- Make sure you have enough processes
  - Enough >> 7

Let the scheduler place processes on SPEs
- Each SPE schedules locally

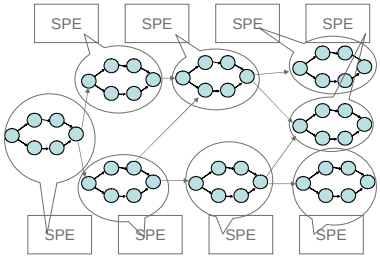
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CSP



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

CSP

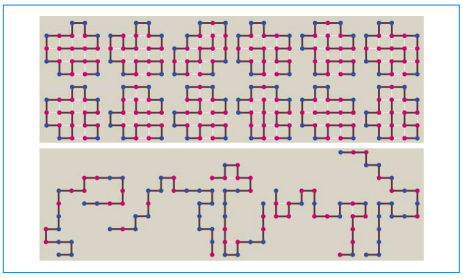


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Performance

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

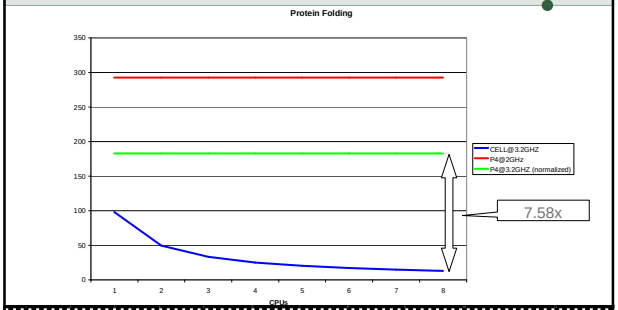
Protein Folding



distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

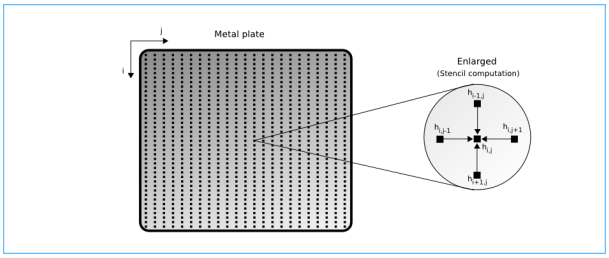
CELL Performance Protein folding



distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

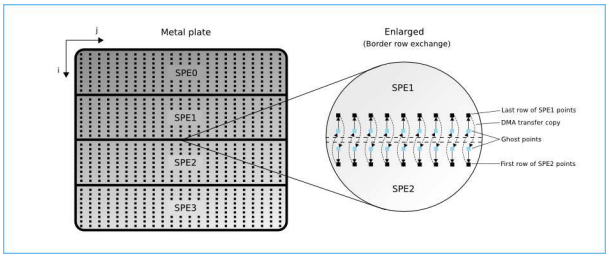
Successive Over Relaxation



distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

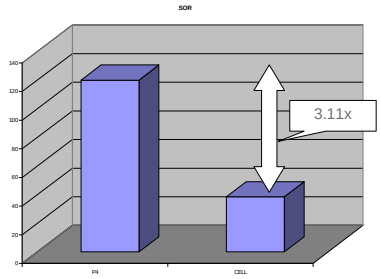
Successive Over Relaxation

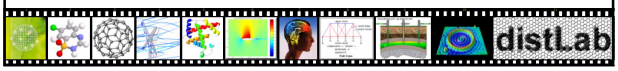


distLab

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Double precision not as impressive





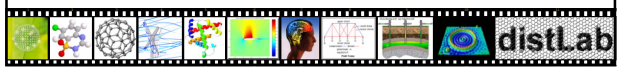
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Conclusions

Next generation hardware is going to be really cool

It's also going to be much harder to program efficiently

Control is going to return to the programmer



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF COPENHAGEN

Thanks to

IBM for a fruitful collaboration

Mohammad Jowkar for doing all the actual work with evaluating the CELL processor and many of the figs