Why Computing in Space?

- Reducing costs
- Dealing with insane amounts of data (Big Data)
- Increasing speed (latency and throughput)
- Supporting growth at a very large scale
- Increasing competitive advantage
- Doing something that could not be done before…

- Cloud Computing!!

“In the cloud, nobody knows that you’re a DFE…”
What is Computing in 3D Space

- First, fix the size of the computer in 3D space
- Then program a CPU machine and a Spatial Computer of the same size, and see the gain from computing in space
- Find a large enough dataset to fill the DFE machine
Computing in Space (1992)
Digital Equipment Corporation DEC PeRLe-1

Applications:
- Long Int. Multiplication
- RSA Cryptography
- Dynamic Programming
- Laplace Heat Equation
- Viterbi Decoder
- Sound Synthesis
- Neural Networks
- Stereo Vision
- Hough Transform
- High Energy Physics
- Image Acquisition
- Wireless LAN testbed
MPC-X 2012 Scalable Dataflow Computing

**MPC-X Architecture**
- Combine CPU and DFE nodes to handle unique compute challenges efficiently
- Use low-latency, high bandwidth Infiniband fabric for expandable compute
- Multi-Scale Cluster optimization balances resources at cluster, rack, node and DFE level

**MPC-X Node**
- 112GB/s Infiniband Connectivity provides exceptional zero-copy bandwidth and latency
- 768GB DRAM to contain massive datasets
- 2GB/s MaxRing for intra-node communication
- PSU redundancy and ‘lights out’ management

1U form “pizza box” form factor
# OpenSPL enabled optimizations

<table>
<thead>
<tr>
<th>Multiple scales of computing</th>
<th>Important features for optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>complete system level</strong></td>
<td>⇒ balance compute, storage and IO</td>
</tr>
<tr>
<td>parallel node level</td>
<td>⇒ maximize utilization of compute and interconnect</td>
</tr>
<tr>
<td>microarchitecture level</td>
<td>⇒ minimize data movement</td>
</tr>
<tr>
<td>arithmetic level</td>
<td>⇒ tradeoff range, precision and accuracy = discretize in time, space and value</td>
</tr>
<tr>
<td>bit level</td>
<td>⇒ encode and add redundancy</td>
</tr>
<tr>
<td>transistor level</td>
<td>⇒ create the illusion of ‘0’ and ‘1’</td>
</tr>
</tbody>
</table>

And more, e.g., trade Communication (Time) for Computation (Space)
### DSF Pricing Table

<table>
<thead>
<tr>
<th>CME Ticker</th>
<th>Bloomberg Ticker</th>
<th>Price</th>
<th>Coupon</th>
<th>PV01</th>
<th>NPV</th>
<th>Implied Rate</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1UM4 2Y</td>
<td>CTPM4</td>
<td>100'057</td>
<td>0.750%</td>
<td>$19.97</td>
<td>$179.69</td>
<td>0.6600%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
<tr>
<td>F1UM4 5Y</td>
<td>CFPM4</td>
<td>100'115</td>
<td>2.000%</td>
<td>$48.49</td>
<td>$359.38</td>
<td>1.9259%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
<tr>
<td>N1UM4 10Y</td>
<td>CNPM4</td>
<td>100'225</td>
<td>3.000%</td>
<td>$90.16</td>
<td>$703.12</td>
<td>2.9220%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
<tr>
<td>B1UM4 30Y</td>
<td>CBPM4</td>
<td>102'770</td>
<td>3.750%</td>
<td>$195.07</td>
<td>$2,843.75</td>
<td>3.6042%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
<tr>
<td>T1UU4 2Y</td>
<td>CTPU4</td>
<td>100'085</td>
<td>1.000%</td>
<td>$19.93</td>
<td>$265.62</td>
<td>0.8668%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
<tr>
<td>F1UU4 5Y</td>
<td>CFPU4</td>
<td>100'110</td>
<td>2.250%</td>
<td>$48.27</td>
<td>$343.75</td>
<td>2.1788%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
<tr>
<td>N1UU4 10Y</td>
<td>CNPU4</td>
<td>101'125</td>
<td>3.250%</td>
<td>$89.55</td>
<td>$1,390.62</td>
<td>3.0948%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
<tr>
<td>B1UU4 30Y</td>
<td>CBPU4</td>
<td>106'020</td>
<td>4.000%</td>
<td>$193.47</td>
<td>$6,062.50</td>
<td>3.6868%</td>
<td>4:00:03 PM CT 4/4/2014</td>
</tr>
</tbody>
</table>

Quotes and analytics are updated every 15 minutes.

### Performance Comparison Table

<table>
<thead>
<tr>
<th>Instrument</th>
<th>CPU 1U-Node</th>
<th>Max 1U-Node</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>European Swaptions</td>
<td>848,000</td>
<td>35,544,000</td>
<td>42x</td>
</tr>
<tr>
<td>American Options</td>
<td>38,400,000</td>
<td>720,000,000</td>
<td>19x</td>
</tr>
<tr>
<td>European Options</td>
<td>32,000,000</td>
<td>7,080,000,000</td>
<td>221x</td>
</tr>
<tr>
<td>Bermudan Swaptions</td>
<td>296</td>
<td>6,666</td>
<td>23x</td>
</tr>
<tr>
<td>Vanilla Swaps</td>
<td>176,000</td>
<td>32,800,000</td>
<td>186x</td>
</tr>
<tr>
<td>CDS</td>
<td>432,000</td>
<td>13,904,000</td>
<td>32x</td>
</tr>
<tr>
<td>CDS Bootstrap</td>
<td>14,000</td>
<td>872,000</td>
<td>62x</td>
</tr>
</tbody>
</table>
3D Spatial perspective on OPEX: Measuring Rack Power at the Socket

• Measurement includes:
  – Cooling, A/C
  – Power supplies
  – Rack level power distribution
  – Networking
  – Storage
  – Memory Chips
  – Compute Chips (CPUs and DFEs)
  – all the other stuff inside a rack that needs electricity

• Power per rack is nice but really we want *Useful Computations per Watt* of rack power consumption
A rack holds computing units of 1U, 2U, 3U, 4U

1U = 19inch \times 36.5inch \times \textbf{1.75inch}
2U = 19inch \times 36.5inch \times \textbf{3.5inch}

Each compute unit has its own power supply

1U CPU servers can have 1-2 mother boards each with multiple CPU chips each with multiple cores.

1U DFE boxes can have 8 DFEs or [6 DFEs + 2 CPU chips], connected via Infiniband

Each 1U DFE box has 384GB of LMEM and 48 independent memory channels.
Spatial perspective on OPEX:  
*Measuring the 3D Space for a program*

1. **Need a good reference point:**
   - Running multithreaded programs on multiple cores, how many cores are there in a 1U box? How many DFEs?
2. **How many SysAdmins do you need per rack?**
3. **Do you measure computations per rack or TeraBytes processed per rack per second?**

*Evaluate performance by comparing 1U to 1U!*
Imaging Platform Example: Weather

1U CPU Node
Wall Clock Time: 2 hours

1U Dataflow Node
less than 2 minutes

Problem size: (Longitude) 13,600 Km x (Latitude) 3330 Km
Simulation of baroclinic instability after 500 time steps.

What does 60x mean?

- Putting a supercomputer into a drone with 60x less space/power consumption.
- 6 days => 2.4 hours computing
- 10 images assembled => 600 images assembled
- 1 Mega Watt => 17 Kilo Watts
- Every $1M spent on buying a dataflow machine delivers power savings over 3 years worth more than $900K.
Economics of Computing Space

you got to have a lot of data.

Total Cost of Ownership = \( f(\text{HW, SW, maintenance, power, real estate, }\ldots) \)

Position of break-even point depends on required programming effort

Number of Runs

Total Cost Reduction

Software Porting Cost
### Total Cost of Ownership = CAPEX + OPEX

Capital Expenditure + Operating Expenditure

<table>
<thead>
<tr>
<th>50x</th>
<th>30x</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>50x</strong> Speed-up per 1U server node</td>
<td>• <strong>30x</strong> Speed-up per 1U server node</td>
</tr>
<tr>
<td>• <strong>32</strong> MPC-X Node Solution</td>
<td>• <strong>40</strong> MPC-X Node Solution</td>
</tr>
<tr>
<td>• Equivalent to <strong>1600</strong> CPU-only Nodes</td>
<td>• Equivalent to <strong>1200</strong> CPU-only Nodes</td>
</tr>
<tr>
<td>• <strong>$3.2m</strong> Operational cost savings over 3 years</td>
<td>• <strong>$1.8m</strong> Operational cost savings over 3 years</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>20x</th>
<th>40x</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>20x</strong> Speed-up per 1U server node</td>
<td>• <strong>40x</strong> Speed-up per 1U server node</td>
</tr>
<tr>
<td>• <strong>50</strong> MPC-X Node Solution</td>
<td>• <strong>32</strong> MPC-X Node Solution</td>
</tr>
<tr>
<td>• Equivalent to <strong>1000</strong> CPU-only Nodes</td>
<td>• Equivalent to <strong>1280</strong> CPU-only Nodes</td>
</tr>
<tr>
<td>• <strong>$1.7m</strong> Operational cost savings over 3 years</td>
<td>• <strong>$2.6m</strong> Operational cost savings over 3 years</td>
</tr>
</tbody>
</table>
Example of Economics of Computing in Space

On the plus side:

1. Budget for buying a new machine is $2.8M
2. An x86 CPU-only machine for $2.8M has $1.3M annual electricity cost
3. A DFE machine for $2.8M creates $27K annual electricity cost

Given that the purchase budget is fixed, the decision to use DFEs saves $1.27M per year.

AND the DFE machine is a lot faster than the CPU machine with the same purchase price.

So by using DFEs, it is possible to get the advantage of saving operational costs AND at the same time get a machine that is a lot faster.

On the minus side:

Have to port applications to DFEs, and operational savings might not be attractive to everyone...
The management challenges

• Resource Utilization
  – CPU bottleneck may mean no work for DFE
  – Direct impact on <X>/performance metrics

• Varying workloads
  – Some applications may need many DFEs, others few or none

• Context switching
  – DFEs are configured for a particular task
  – Context switching is expensive and should be minimized

Enable system to *adapt* to runtime workload
DFE Cluster Management

- ClusterMap job distribution for DFE clusters
- Optimized for distributing large numbers of small compute jobs with complex dependencies
Rack-level optimization

- Optimized to balance resources for particular application challenges
- Flexible at design-time and at run-time
Using a single DFE with SLiC

• **Simple Live CPU Interface:**
  MaxCompiler-generated + fixed software functions for interacting with DFEs

```c
#include auto-generated AVG_* prototypes
#include “AVG.h”

// Load .max-file onto any (“*”) available engine
max_file_t *avg_maxfile = AVG_init();
max_engine_t *eng1 = max_load(maxfile, “*”);

// Set-up and execute an “action”
float *x = <relevant data>, *y = <relevant data>;
AVG_action_t a;
a.instream_x = x;
a.outstream_y = y;
a.total_items = n;
AVG_run(eng1, &a);

// Shutdown
maxUnload(eng1);
```
Virtual DFEs

• Aggregate zero or more physical DFEs into *virtual DFEs*
• Multiple CPU clients can share virtual DFEs
  – Minimize reconfiguration
  – Allow sharing of datasets
• Every thread/process wishing to use the same group uses a common “tag”
• SLiC automatically allocates DFEs for group
Accessing a Virtual DFE group

```c
max_file_t *maxfile = AVG_init();

// Load a named group of engines that will be shared with other processes.
// All processes in group must make same call.
int group_size = 10;
max_group_t *grp =
    max_load_group(maxfile, MAXPROP_SHARED, "tag@*", group_size);

// ... sometime later ...

// Set-up actions as normal and then either
AVG_action_t *act1 = <relevant data>, *act2 = <relevant data>;
if (condition) {
    // Lock DFE to preserve state
    max_engine_t *eng = max_lock_any(grp);
    AVG_run(eng, &act1);
    AVG_run(eng, &act2);
    max_unlock(eng);
} else {
    // DFE run atomically (fast)
    AVG_run_group(grp, &act1);
    AVG_run_group(grp, &act2); // no state preserved between calls
}
```
Virtual/Physical Resource Balancing

- Type A tasks
- Group of DFEs
- Multiple physical resources become one virtual resource

DFE Scheduling

Type A tasks

DFE Migration to Group

Type B tasks
Architecture key points

• Action dispatch is fast & distributed
• Three levels of load balancing:
  – Actions dispatched to which MPC-X
  – Actions dispatched to which DFE
  – Physical DFEs (re)allocated between virtual DFEs
• Cluster configuration can be continually adapted at runtime to meet latest resource demands
Case Study: Geoscience Application

- Original runs on high end 16-core x86 servers
- Highly optimized production code

1..10000 jobs

Static data A (8GB)

Phase 1 (400s)

Temp data (Many GB)

Static data B (multi-TB)

Select partial state to update

Subset of B ~400GB

Phase 2 (1800s)

Only ~50GB changes per phase 2
Dataflow Implementation: MPC-C

- Reload DFEs between phase 1 and phase 2
Runtime impact: MPC-C with 4 DFEs

<table>
<thead>
<tr>
<th></th>
<th>DFE Potential</th>
<th>Actually achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>19x</td>
<td>10x</td>
</tr>
<tr>
<td>Phase 2</td>
<td>20x</td>
<td>15x</td>
</tr>
<tr>
<td>Overall</td>
<td>~20x</td>
<td><strong>11x</strong></td>
</tr>
</tbody>
</table>

- DFEs are under-utilized
- Disk + CPU limited
- 17% of time in initialization
DFE under-utilization

- Increase CPU power per node?
  - 20% faster clock frequency possible but would still be limiting and costs a lot of power

- Increase disk speed per node?
  - Already using high speed disks, not possible to add more drives in the form factor

- Switch to SSDs
  - Continuous write-read cycle poor for SSD endurance

- Decrease number of DFEs per node
  - “Ideal” number of DFEs differs for different phases & based on individual job characteristics
Dataflow Implementation: MPC-X

- Don’t reconfigure DFEs
- Allocate N DFEs per CPU node for Phase 2
- Allocate M DFEs per cluster for Phase 1
- Vary N, M as workload characteristics change
- Overlap Phase 1 of job J with Phase 2 of job J-1
Runtime impact: MPC-X

- Performance set by speed of phase 2 computation, phase 1 requires only ~0.6 DFE
- More CPUs accompany each MPC-X, but each node is less expensive (less memory needed)

<table>
<thead>
<tr>
<th>Table</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU nodes per MPC-X</td>
<td>2.2</td>
</tr>
<tr>
<td>Speedup per MPC-X</td>
<td>38.9x</td>
</tr>
</tbody>
</table>
Summary

• Optimizing utilisation is key to maximizing performance
• Sharing physical resources can improve runtime
• Intelligent resource balancing can maximize overall system output
• Intelligent task dispatching should consider resource heterogeneity