CO405H

Computing in Space with OpenSPL
Topic 14: 3D Space (Part 2)
Virtualization of Dataflow Engines

Oskar Mencer **Georgi Gaydadjiev**

Department of Computing
Imperial College London

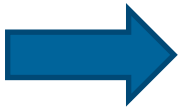
http://www.doc.ic.ac.uk/~oskar/
http://www.doc.ic.ac.uk/~georgig/

CO405H course page: http://cc.doc.ic.ac.uk/openspl14/
WebIDE: http://openspl.doc.ic.ac.uk
OpenSPL consortium page: http://www.openspl.org

o.mencer@imperial.ac.uk g.gaydadjiev@imperial.ac.uk

The Challenge

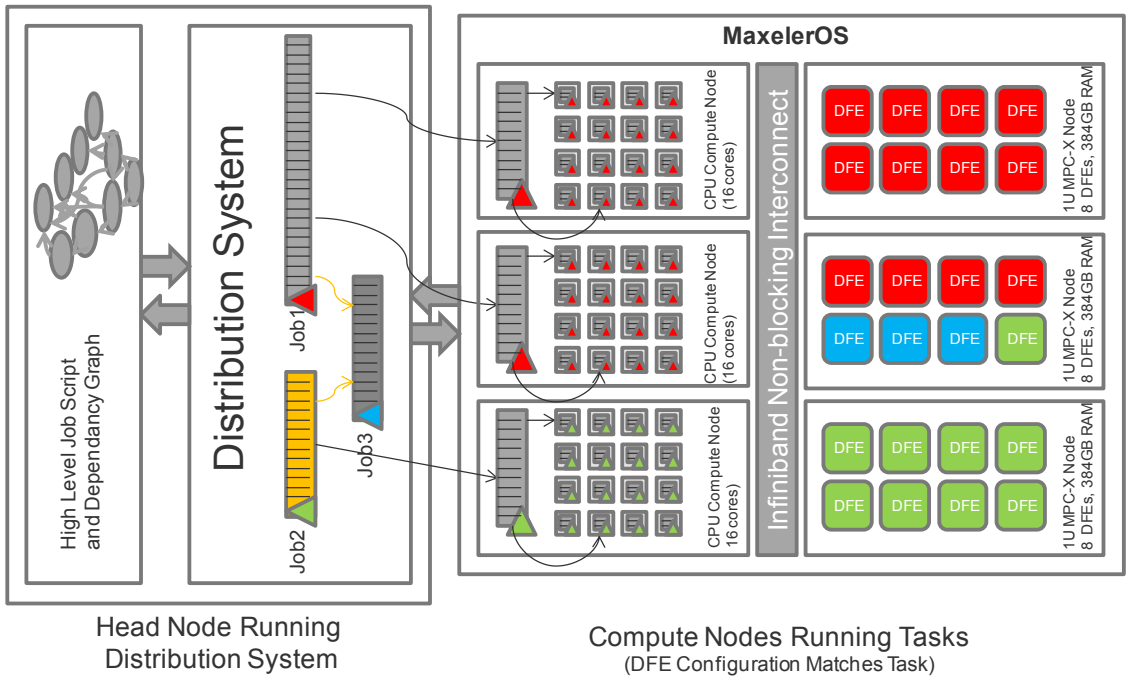
- Resource Utilization
  - CPU bottleneck may mean no work for DFE
  - Direct impact on <X>/performance metrics
- Varying workloads
  - Some applications may need many DFEs, others few or none
- Context switching
  - DFEs are *configured* for a particular task
  - Context switching is expensive and should be minimized



Enable system to *adapt* to runtime workload

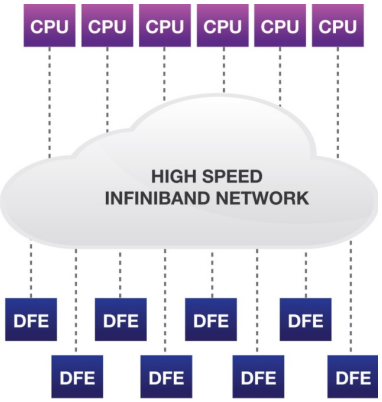
DFE Cluster Management

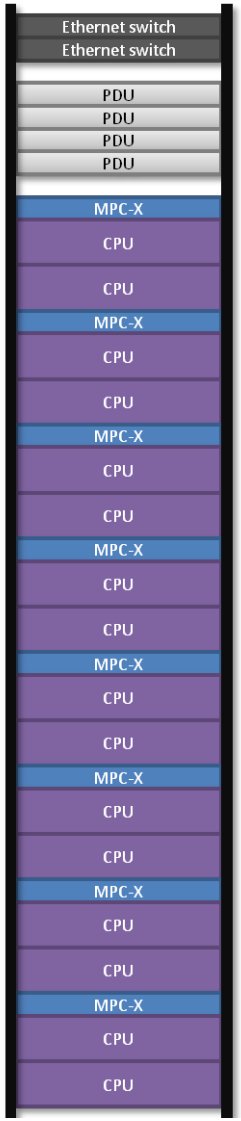
- ClusterMap job distribution for DFE clusters
- Optimized for distributing large numbers of small compute jobs with complex dependencies



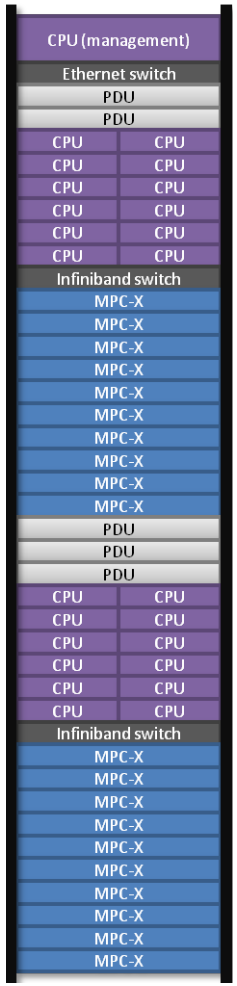
Rack-level optimization

- Optimized to balance resources for particular application challenges
- Flexible at design-time and at run-time





48U seismic imaging cluster



42U in-memory analytics cluster

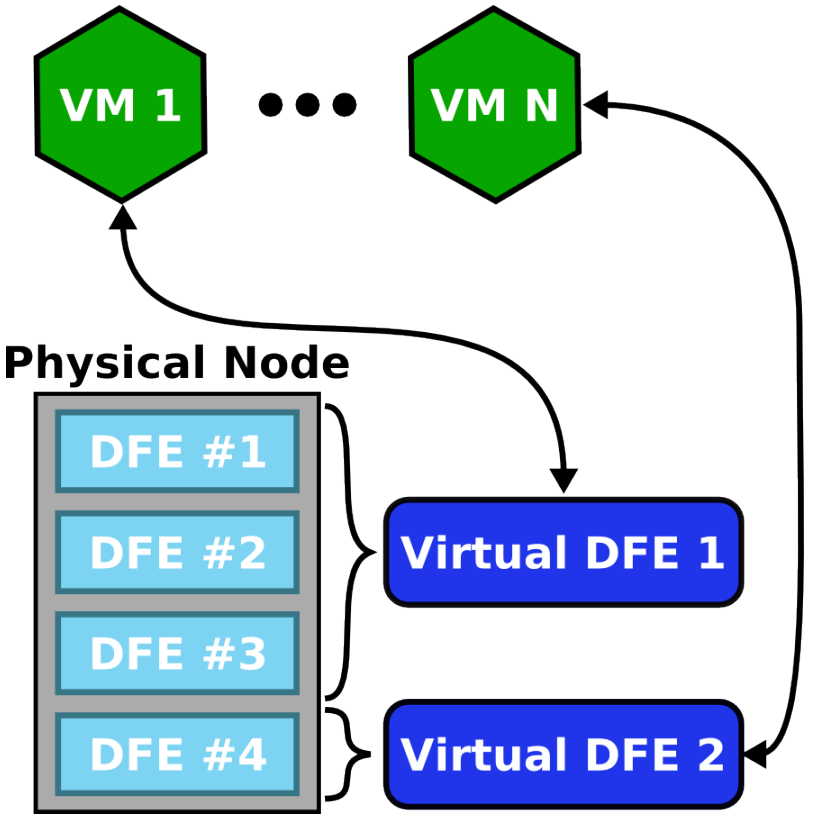
Using a single DFE with SLiC

- **S**imple **Li**ve **C**PU Interface:
  MaxCompiler-generated + fixed software functions for interacting with DFEs

```
//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
max_unload(eng1);
```

Virtual DFEs

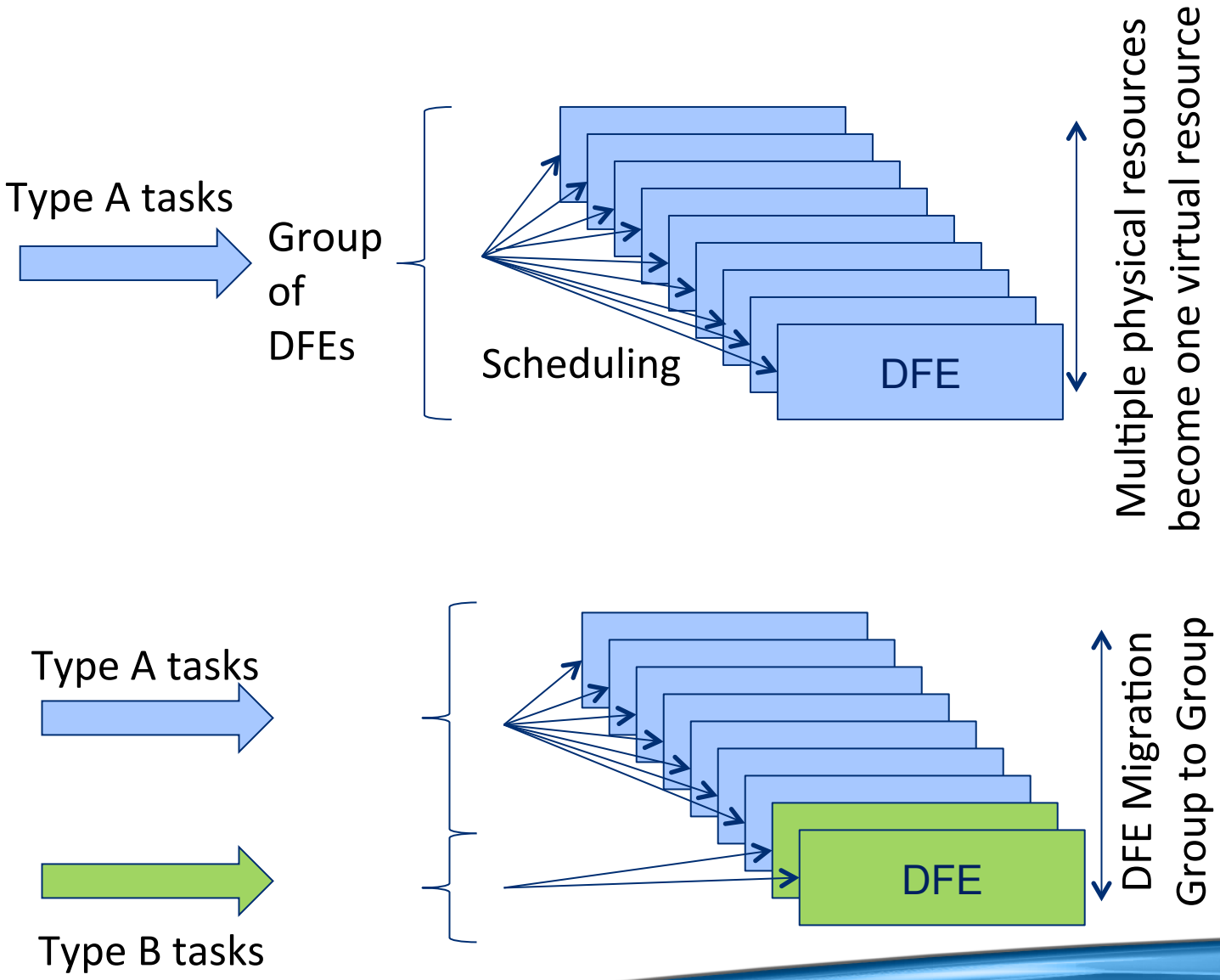
- Aggregate zero or more physical DFEs into *virtual DFEs*
- Multiple CPU clients can share virtual DFEs
  - Minimize reconfiguration
  - Allow sharing of datasets
- Every thread/process wishing to use the same group uses a common “tag”
- SLiC automatically allocates DFEs for group

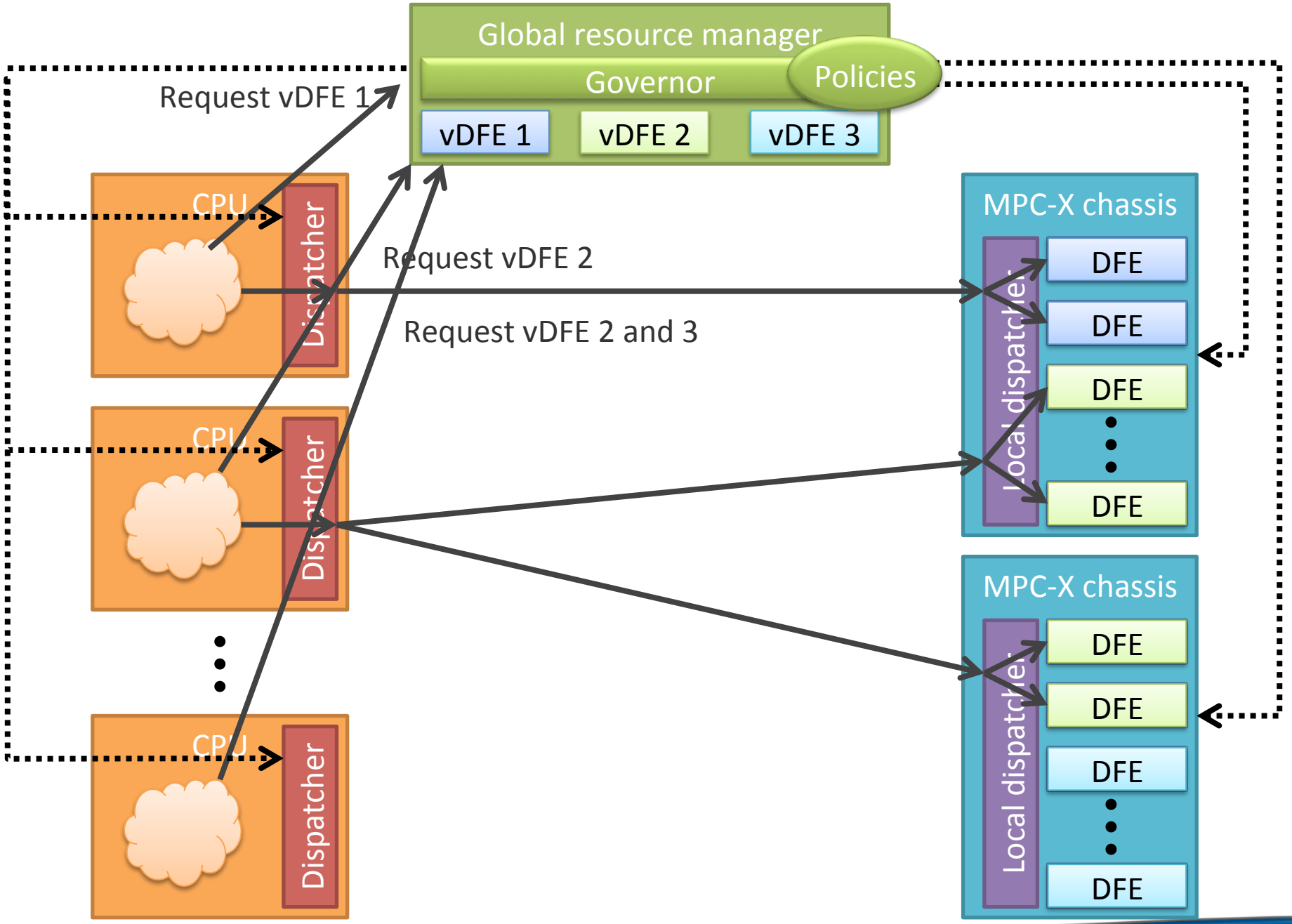


Accessing a Virtual DFE group

```
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(grp1);
    AVG_run(eng, &act1);
    AVG_run(eng, &act2);
    max_unlock(eng1);
} else { // DFE run atomically (fast)
    AVG_run_group(grp, &act1);
    AVG_run_group(grp, &act2); // no state preserved between calls
}
```

Virtual/Physical Resource Balancing



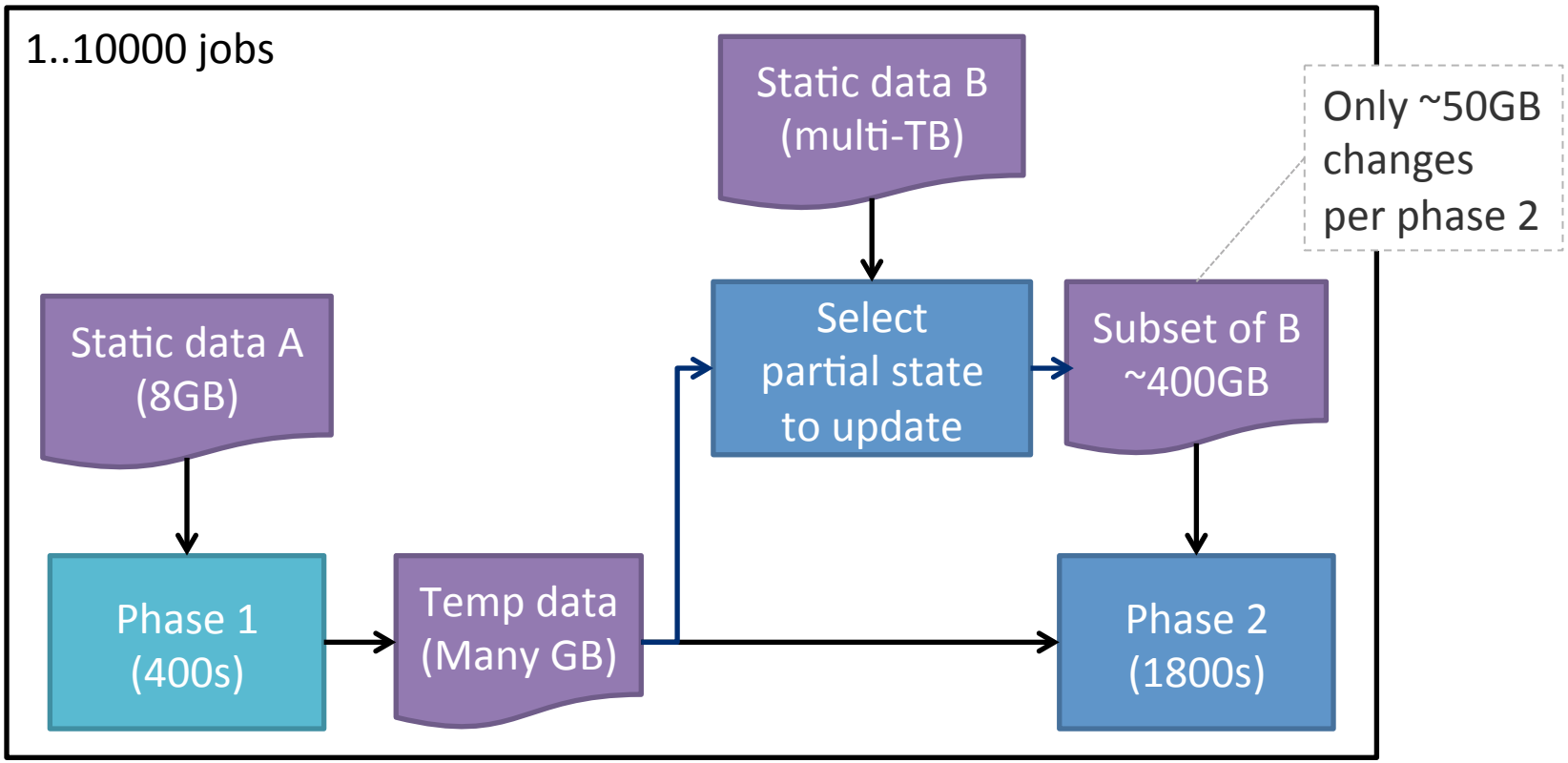

Global resource manager
Governor
Policies
vDFE 1
vDFE 2
vDFE 3
Request vDFE 1
Request vDFE 2
Request vDFE 2 and 3
CPU
CPU
CPU
Dispatcher
Dispatcher
Dispatcher
MPC-X chassis
Local dispatcher
DFE
DFE
DFE
DFE
MPC-X chassis
Local dispatcher
DFE
DFE
DFE
DFE

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

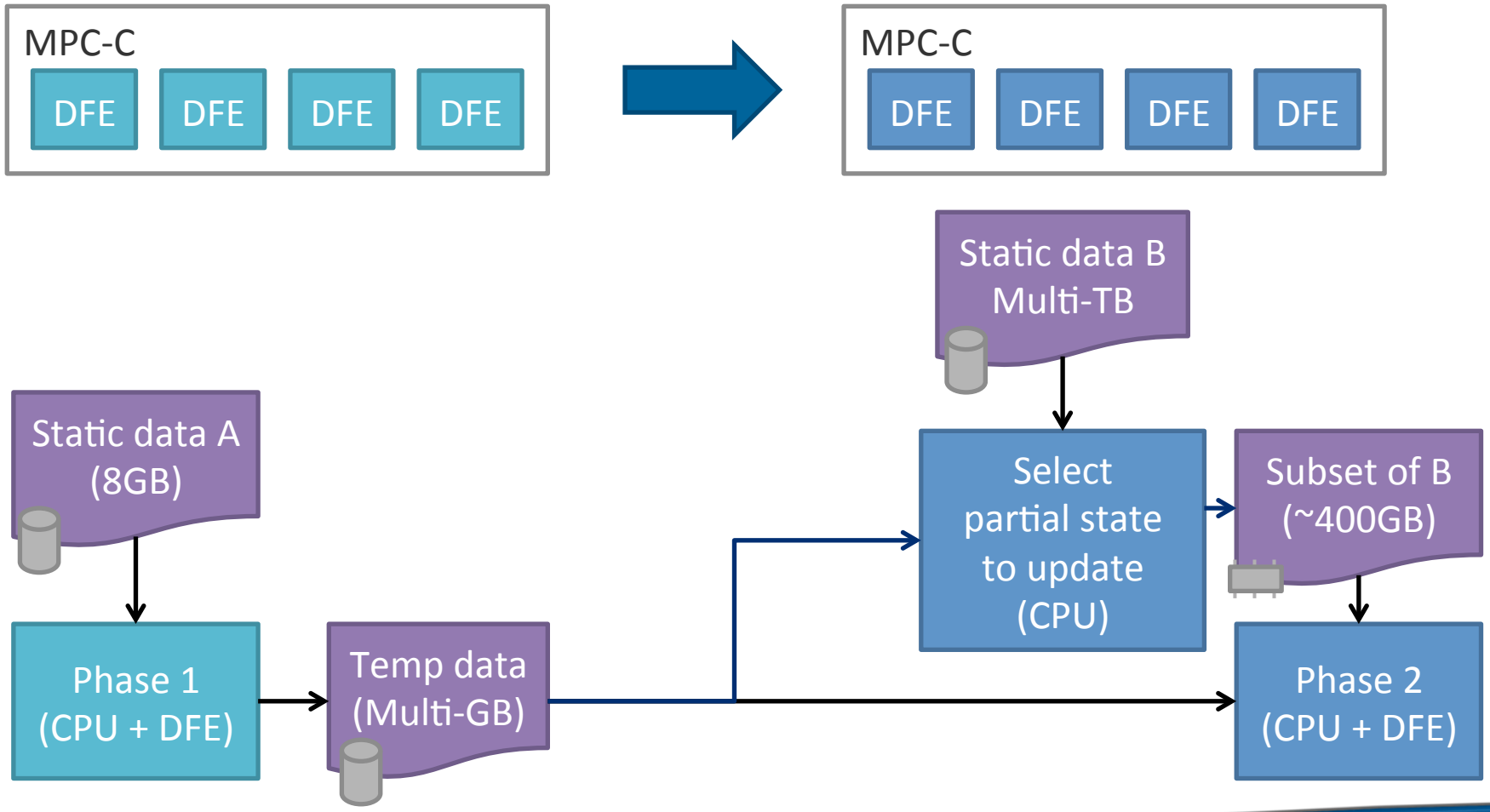
Case Study: Geoscience Application

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

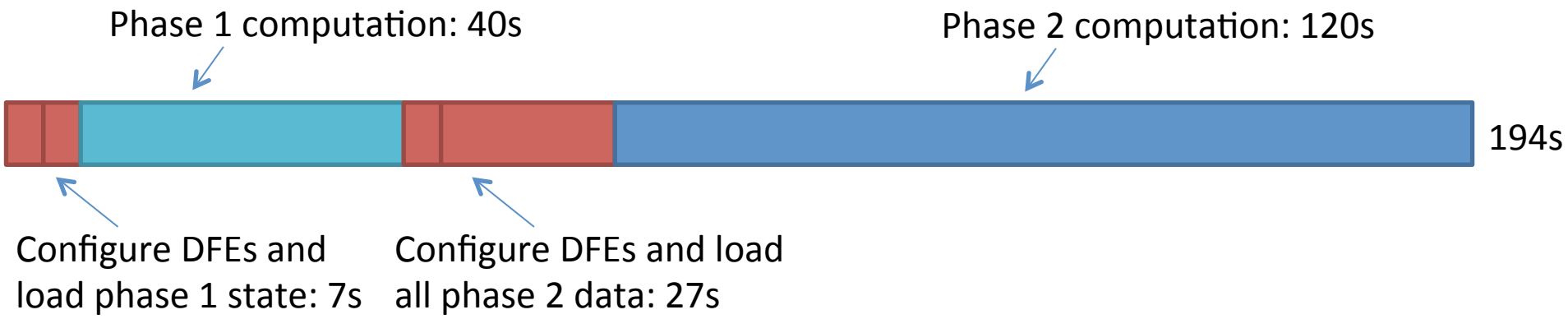


Dataflow Implementation: MPC-C

- Reload DFEs between phase 1 and phase 2



Runtime impact: MPC-C with 4 DFEs



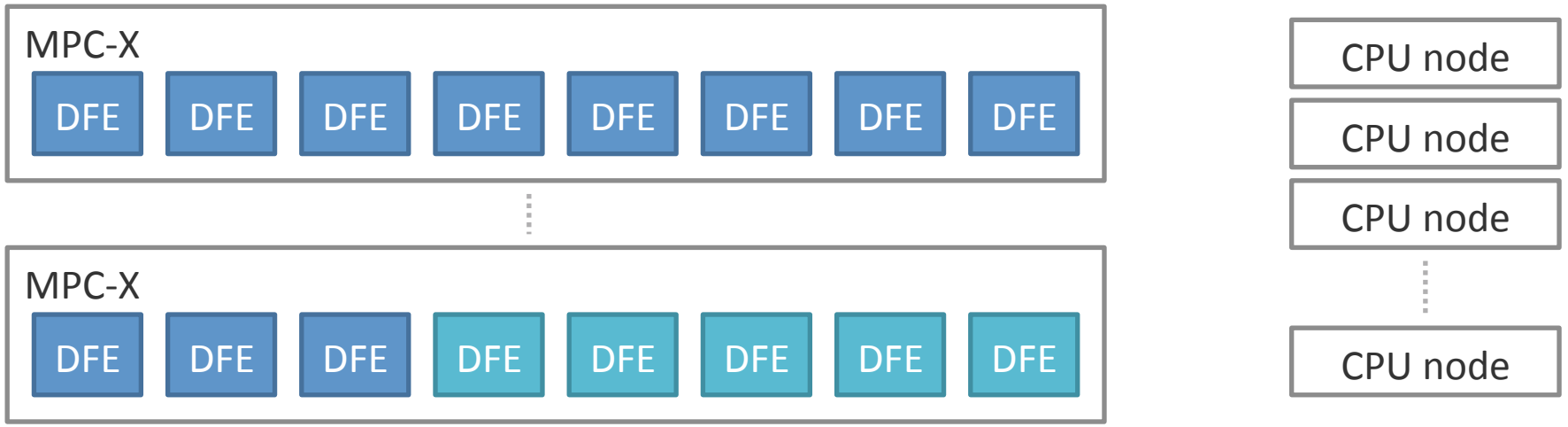
- DFEs are under-utilized
- Disk + CPU limited
- 17% of time in initialization

	DFE Potential	Actually achieved
Phase 1	19x	10x
Phase 2	20x	15x
Overall	~20x	**11x**

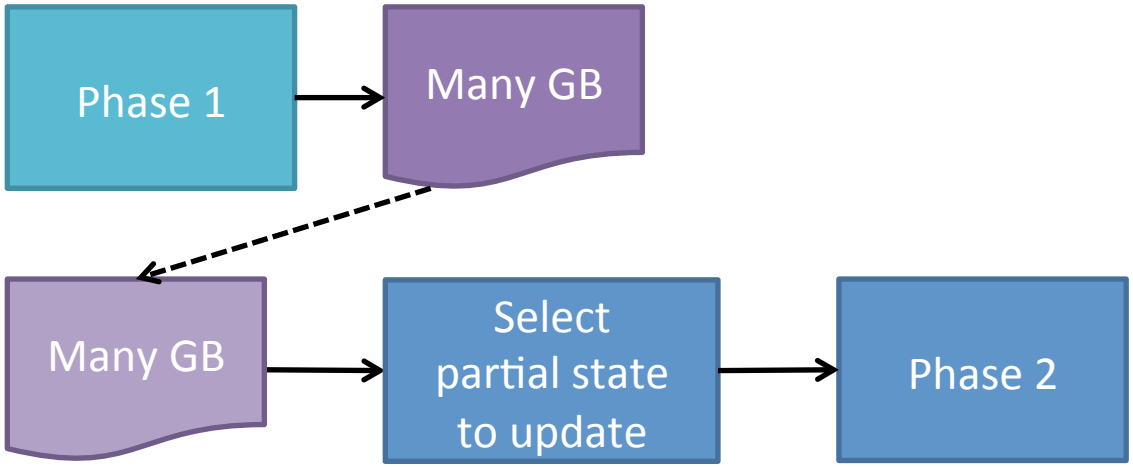
DFE under-utilization

- 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

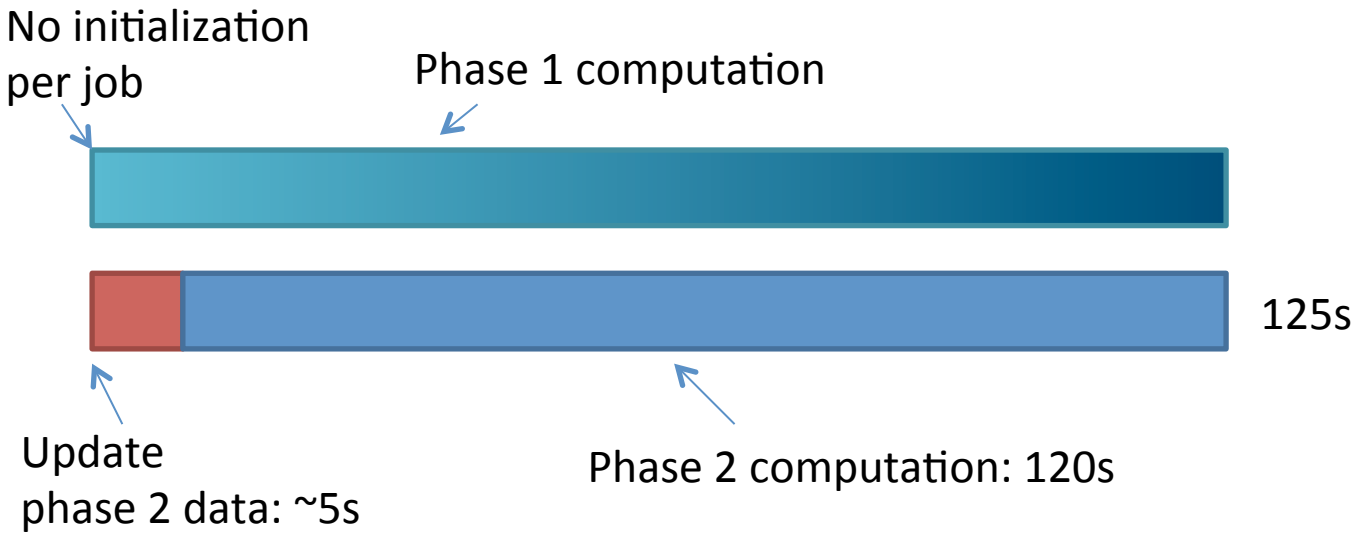
Dataflow Implementation: MPC-X



- 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)

CPU nodes per MPC-X	2.2
Speedup per MPC-X	**38.9x**

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