# Petascale Software Challenges

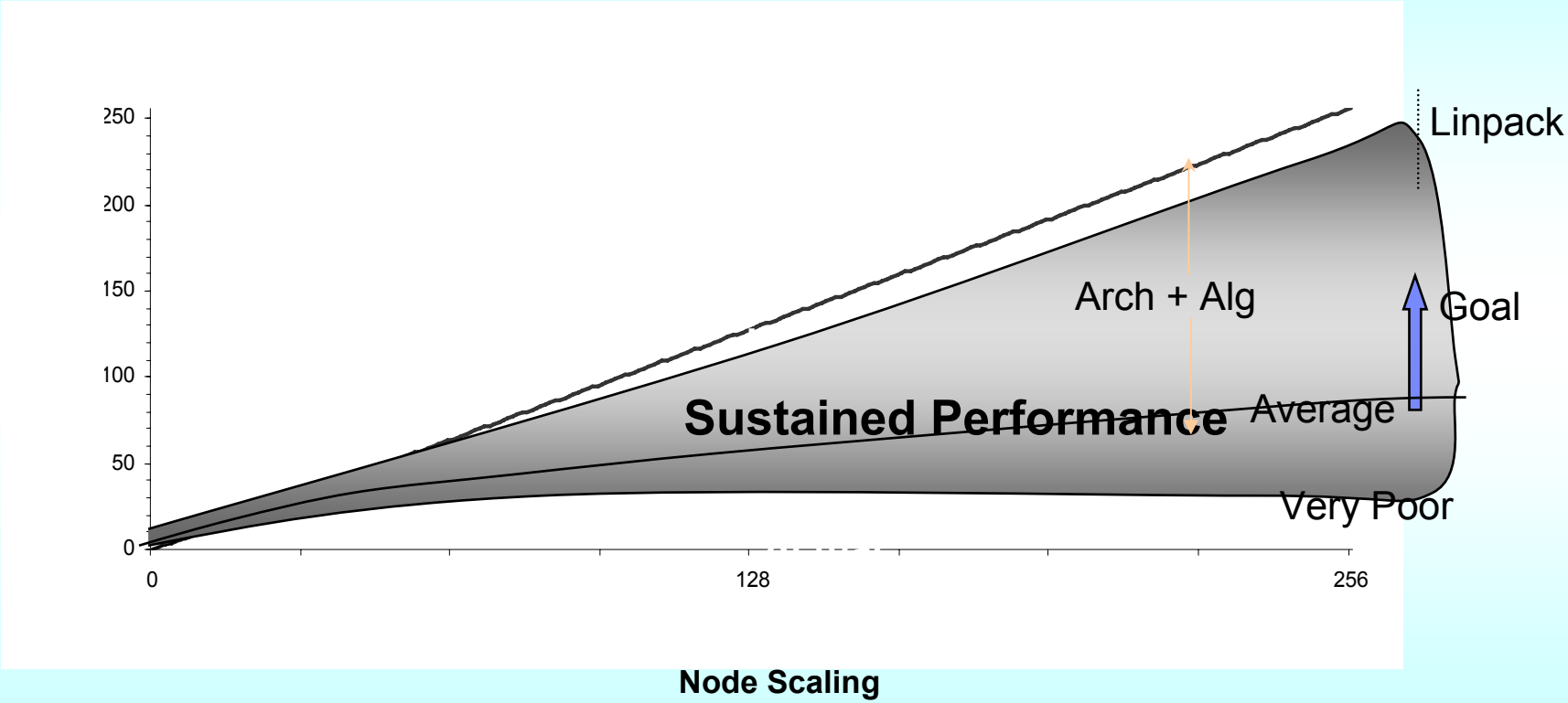
*Piyush Chaudhary*
*piyushc@us.ibm.com*

*High Performance Computing*

# Fundamental Observations

- **Applications are struggling to realize growth in sustained performance at scale**
  - Reasons need to be understood and addressed
- **Productivity enhancements needed to make supercomputing accessible**
- **Domain of operation for the HPC software stack needs to expand beyond a single supercomputer cluster**
  - Domain needs to extend to the **entire data center** (or even the enterprise)
  - Peta-scale systems **co-exist with other large systems** in the data center
  - **Operational efficiency** of Supercomputers often overlooked
- **MTBF: Mean time between failure is increasingly critical at scale**
  - **Continuous operation** – always being able serving applications becomes critical

# Sustained Performance vs. Peak




**Key Problem: Gap between peak and sustained continues to widen**

**Goal: Understand reasons for the gap and attack the problems**

# Ideas to Address Sustained Performance

- **Communication latency:**
  - Burst MMIO
  - Cache injection
  - Lock overhead reduction
  - Integrated I/O, Express and Ethernet closer to the micro-processing elements
- **Overlap computation/communication:**
  - asynchronous data mover
- **Collective Communication overheads:**
  - COE and daemon squashers (co-scheduling)
  - RDMA exploitation
- **Memory latency:**
  - Pre-fetch hints to subsystems and applications (SMT assist threads)
  - Drive towards zero cache miss execution in the latency critical paths (Compiler approaches)
- **Continuous Program Optimization**
- **Algorithmic**
  - Better tools to assist in smarter application development ( Locking & Message passing impacts)

**These are only some of the ideas. Many more being explored**

# Productivity Domains

## Programmer

- Develop Applications
- Debug Applications
- Tune Applications

## System Operational ή

- Maximize System throughput
- Maximize Enterprise ή
- Ensure system balance

## Administrator

- Storage management
- Network management
- Install, upgrades
- System monitoring

## RAS

- Continuous Operation
- Problem isolation
- FFDC
- Serviceability

# Programmer Productivity

- **Developing new parallel applications**
  - IDE environments
  - Libraries
- **Debugging parallel applications**
  - Tracing and parsing hooks
  - Static analysis tools
  - Rational
- **Performance tuning parallel applications**
  - HPC Toolkit, FDPR, compiler feedback, static analysis tools
- **Application/system tuning parameters**
  - CPO

# IDE: Programming Model Specific Editor

Java - test.c -

File Edit Navigate Search Project Run MPI Artifacts Window Help

Java

C/C++ P...

- MyCProject
  - Includes
  - Debug
  - Release
  - test.c
    - mpi.h
    - stdio.h
    - string.h
    - main
    - MPI_myMPIFunctio
- TestIndexer
- TestNoIndexer

test.c

```
        dest = 0;
        /* use strlen+1 so that '\0' get transmitted */
        MPI_Send(message, strlen(message)+1, MPI_CHAR,
            dest, tag, MPI_COMM_WORLD);
    }
    else{
        printf("From process 0: Num processes: %d\n",p);
        for (source = 1; source < p; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, tag
                MPI_COMM_WORLD, &status);
            printf("%s\n",message);
        }
    }
    /* shut down MPI */
    MPI_Finalize();
```

Outline

- stdio.h
- string.h
- mpi.h
- MPI_myMPIFunction
- main

Problems | Javadoc | Declaration | MPI Table View

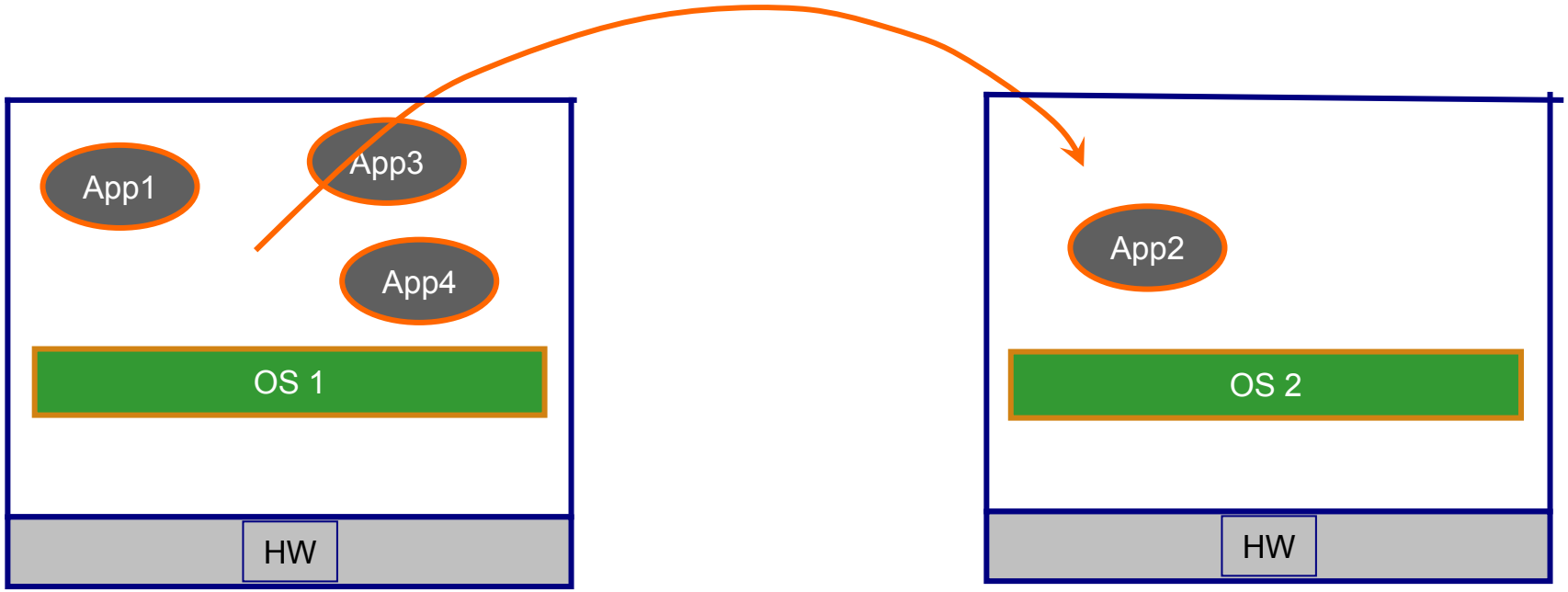
| MPI Artifact | Filename | LineNo | Construct |
|---|---|---|---|
| MPI_Init | test.c | 23 | Function Call |
| MPI_Comm_rank | test.c | 26 | Function Call |
| MPI_Comm_size | test.c | 29 | Function Call |
| MPI_Send | test.c | 37 | Function Call |
| MPI_Recv | test.c | 43 | Function Call |
| MPI_Finalize | test.c | 49 | Function Call |

Writable | Smart Insert | 43 : 21

# Programming Models Support

- **Models supported today**
  - MPI – Two sided: Message Passing Interface
  - LAPI – One sided programming model
    - Platform for middleware and exploring other programming models
  - SHMEM – shared memory programming model
- **Future Models**
  - X10 concepts being applied to HPLS initiatives
  - UPC – Unified Parallel C
- **Component Libraries**
  - ESSL, PESSL enhancements, COIN-OR

# MetaCluster Provides Application Mobility

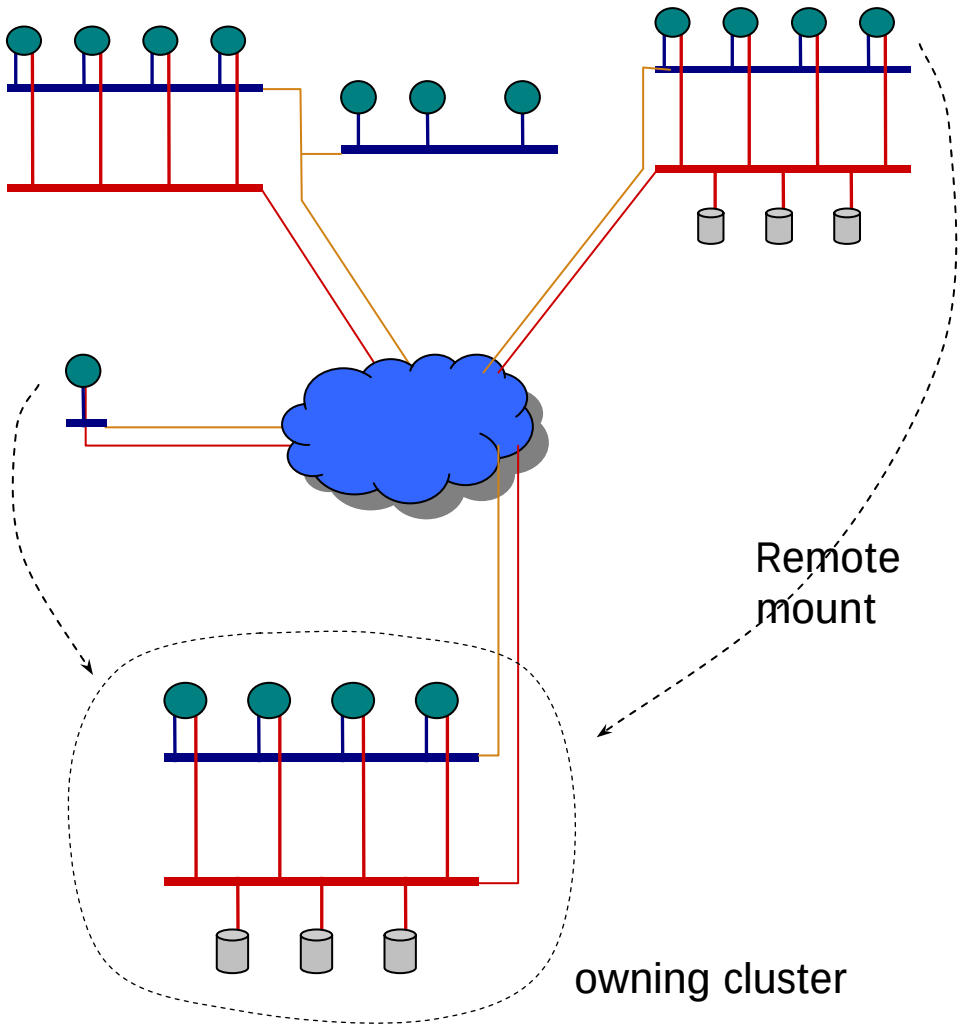



MetaCluster: Application Containers enabling on-demand mobility of workload

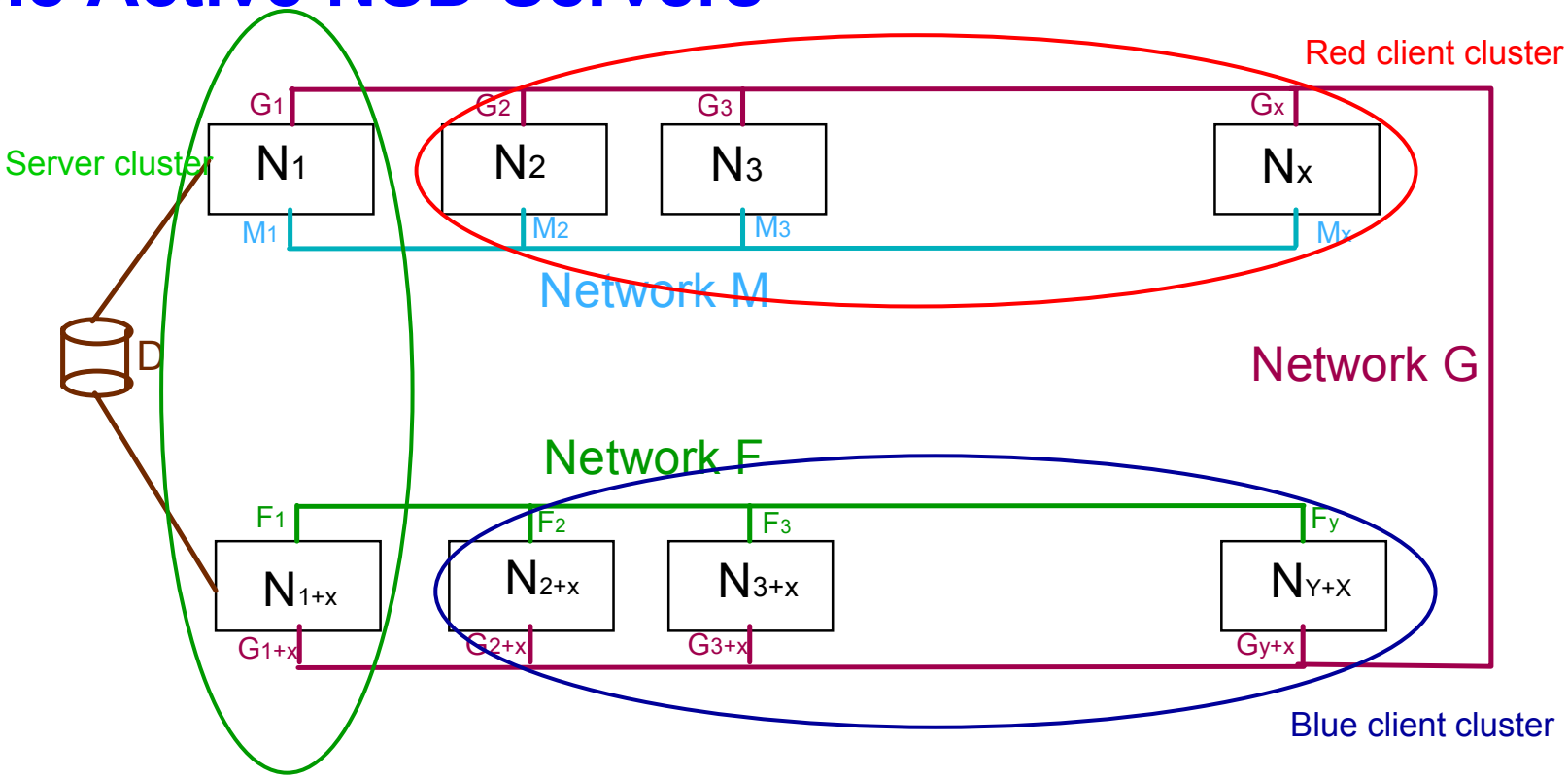
- Lowest overhead of all virtualization technologies (<1%)
- Finest grain: application-level; enables app-centric management
- Largest scope of supported platform (Intel, AMD, Linux, SMP, 32b, 64b)
- Enterprise ready with preservation of all states including TCP
- Metacluster Application Containers compliment Virtual Machines.

# Facility Wide File Sharing

- **Each FS belongs to one "server" cluster, responsible for**
  - providing storage
  - administration
  - lock management
  - recovery
- **"client" nodes can mount FS from server cluster and**
  - request locks
  - access data & metadata directly over the SAN
- **External access also enabled through**
  - NFS, Samba



# Multiple Active NSD Servers




- Client nodes favor NSD servers on the same subnet (presumably the same fabric)
- High availability - when the NSD server for a LUN is not reachable:
  - Fall back to other NSD servers on the same subnet
  - Next, fall back to NSD servers on other routable subnets
- Multi-cluster implications
  - NSD servers for a file sysem must all be in the same cluster
  - If using multiple active NSD servers in multi-cluster environment, configure NSD node on client subnet as a member of one server cluster exporting the file system

# Resource Management Vision

- **Exploit GPFS WAFS, and Meta-cluster capability to make enterprise wide scheduling and RM efficient**
- **Multi-platform**
  - AIX and Linux already available
  - Extending to p (AIX, Linux), x86 Linux, Blue Gene
- **Invest in enhancing LL through advanced technologies**
  - Reservation in advance enhancements
  - Flexible usage model and management for various large page pools
  - Exploitation of SMT in a more flexible fashion
  - Stronger affinity controls and usage limits
  - Faster and scalable job launch
  - Modern GUI with user friendly and intuitive interface

# Example Demonstrating Enterprise Operational Efficiency and the Scale Across Vision

**Without C/R and Relocation**

Cluster 1: 4 jobs running. 2 nodes free

Cluster 2: 2 jobs running, 6 nodes free

Utilization: Cluster 1: 87%, Cluster 2: 63%

**Enterprise Utilization**: 75%

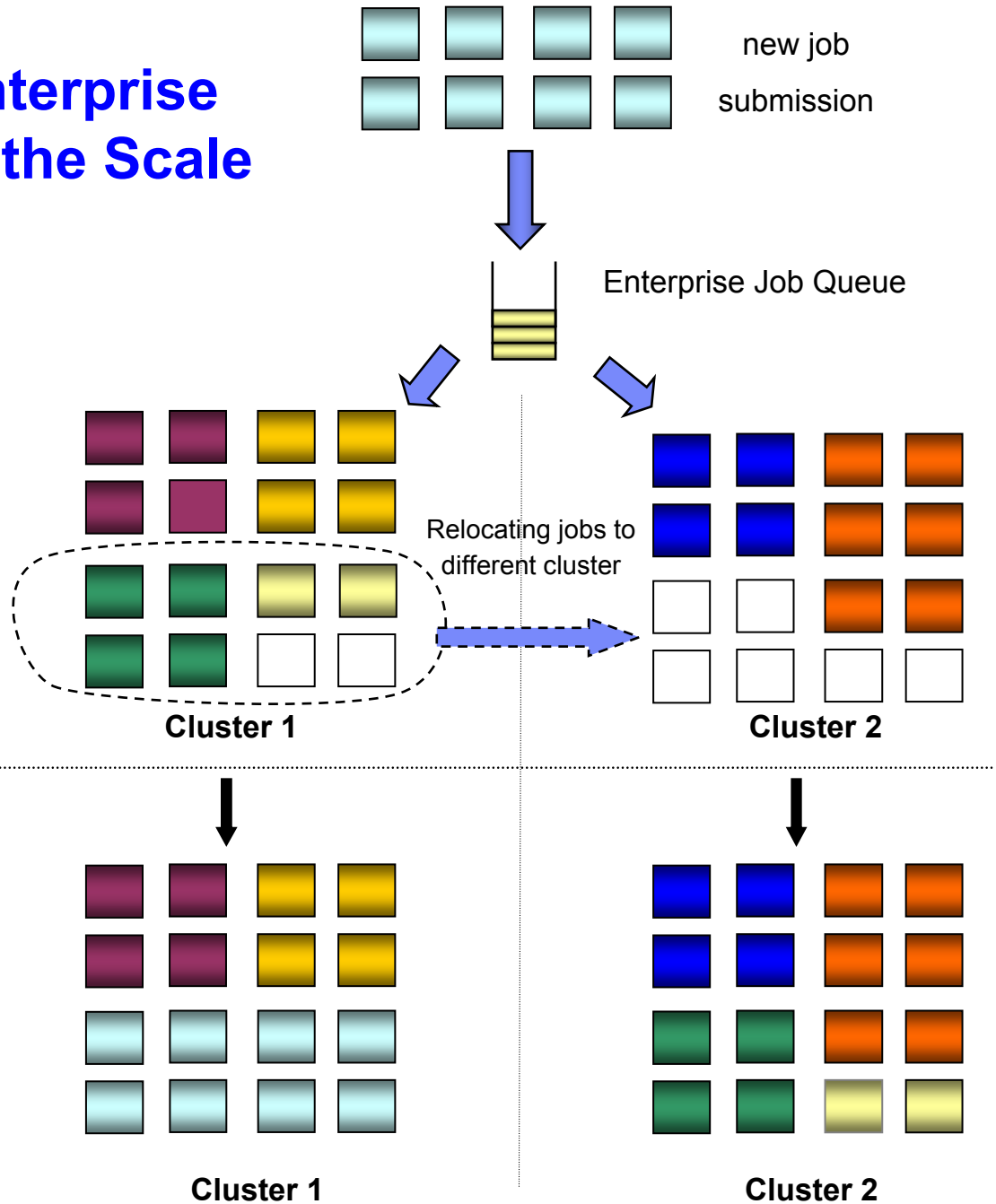
**With C/R and Relocation**

Cluster 1: 3 jobs running. 0 nodes free

Cluster 2: 4 jobs running, 0 nodes free

Utilization: Cluster 1: 100%, Cluster 2: 100%

**Enterprise Utilization**: 100%



# System Operational Efficiency

- **Maximize utilization, and throughput of the systems in the enterprise**
- **Issues are**
  - Applications see inconsistent performance when run on production system
    - Application shares resources with other applications in the system
    - OS jitter impacts application run time especially at scale
  - Ensuring intelligent balance of applications and utilization of various components critical
    - CPU, memory, network, storage
    - Need a workload mix on the system that utilizes all resources in a balanced fashion
  - Increased intelligence for system tuning parameters
    - Continuous program optimization
  - Ability to pre-empt, checkpoint, relocate, applications seamlessly throughout the enterprise
  - Ability to share files seamlessly across the enterprise

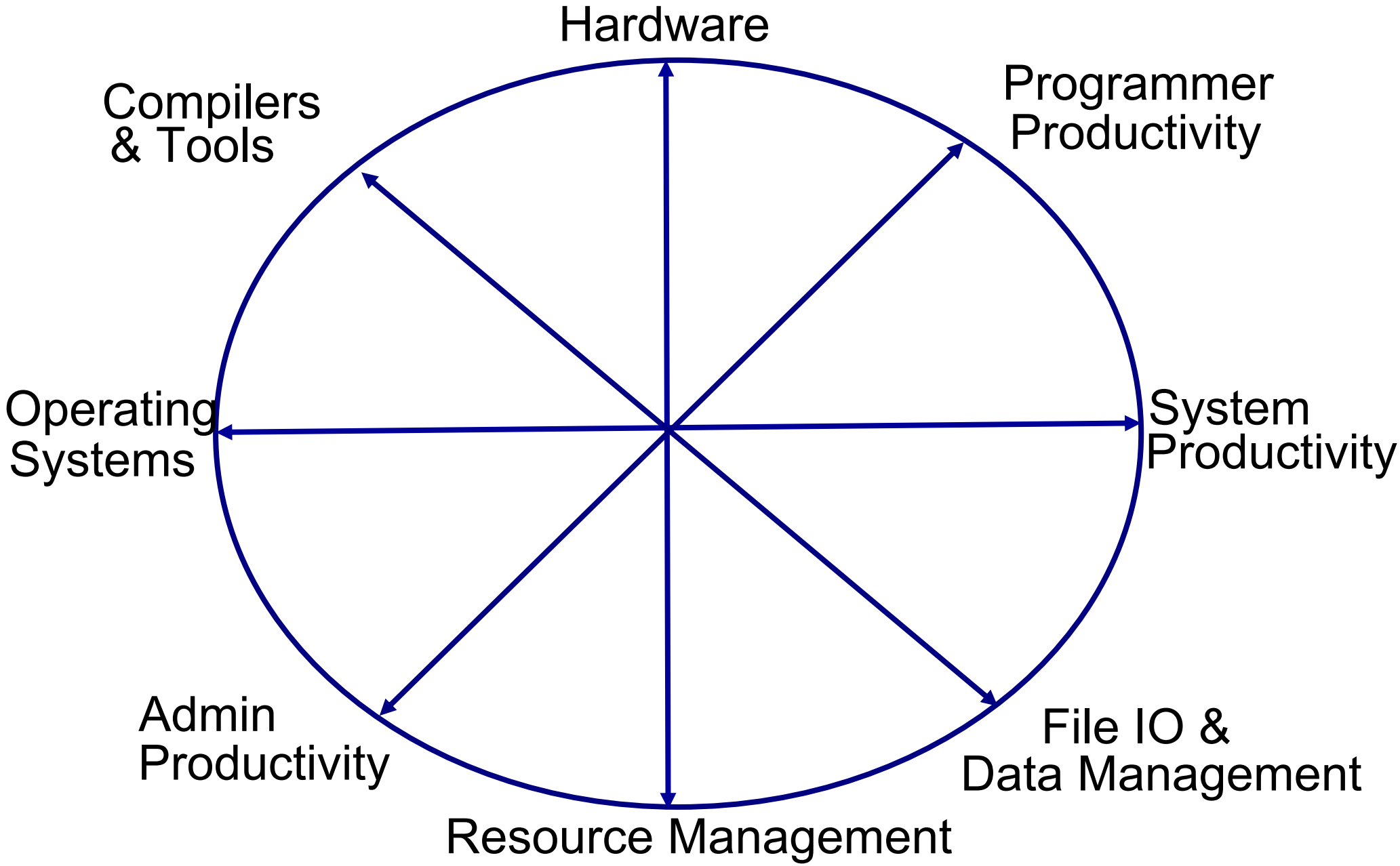
# Systems Management

- **Cluster Install**
  - Robust, fast, and push-button capabilities
  - Concurrent and seamless upgrades
- **Verification capabilities**
- **Monitoring of the network interconnect, power, fans, and other hardware components**
- **Drive towards zero administration and downtime**

# RAS

- **Reliability and Availability**
  - Storage:
    - Controllers, disks, network, IO servers, compute nodes failures tolerated
  - Programming Models
    - Packet drops, link errors, adapter failures, network reach-ability, checkpoint, restart and relocation
  - Resource manager
    - Compute nodes, network and reach-ability challenges
- **Serviceability**
  - Diagnostic tools
  - Monitoring infrastructure, errpts, FSP, trace data, network diagnostics, storage, resource management, systems management

# Balanced System Architecture & Design



# BACKUP