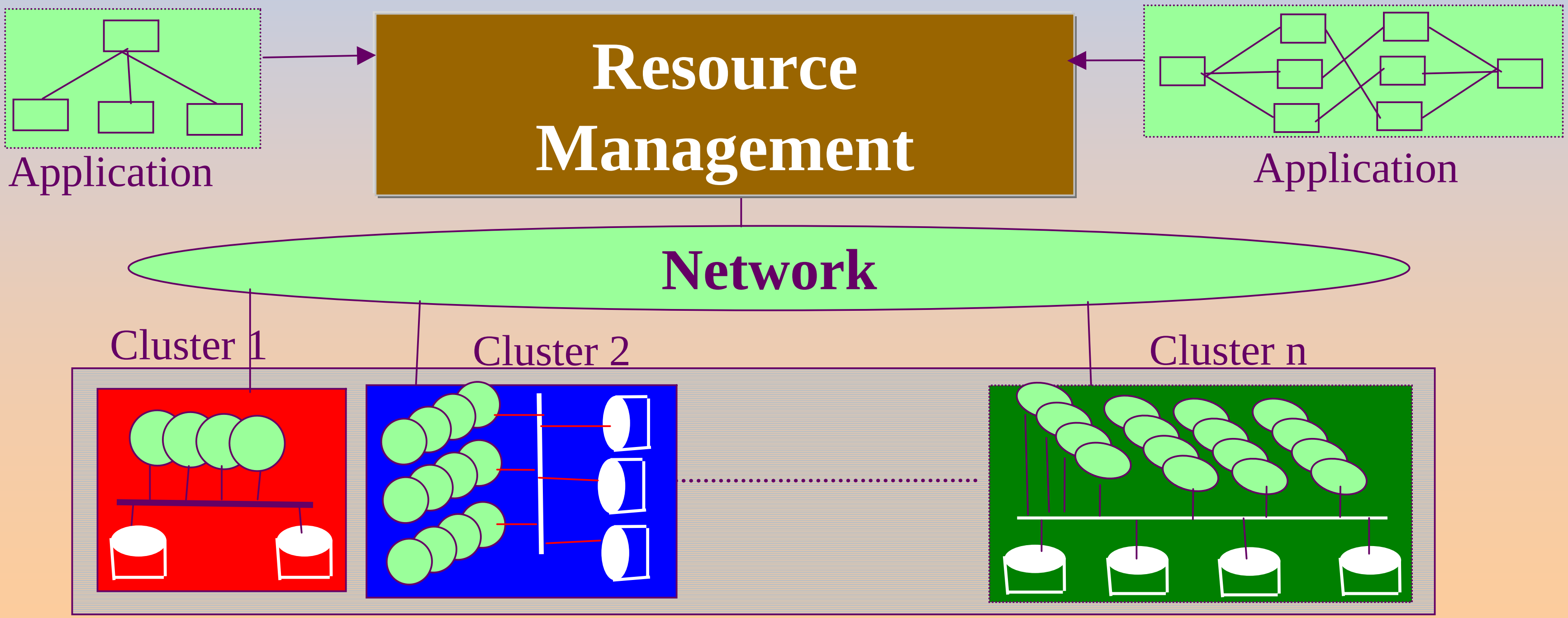
Parallel Jobs Scheduling on Multi-cluster Computing Systems

J. H. Abawajy and S. P. Dandamudi
School of Computer Science
Carleton University, Ottawa, Ontario, Canada
{abawjem, sivarama}@scs.carleton.ca

Background: A cluster

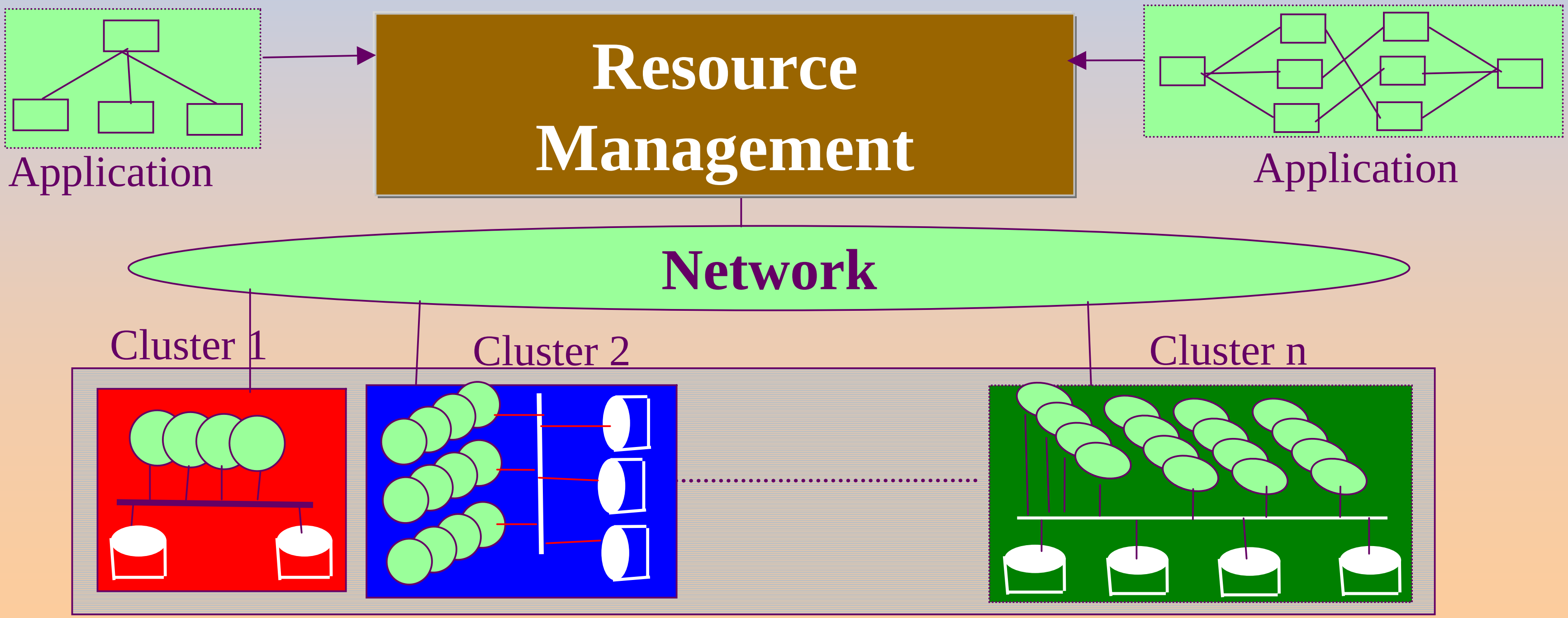
- A cluster is a collection of *n* independent workstations
  - Interconnected by LAN
  - Community-based (i.e., shareable)
  - May be homogeneous or heterogeneous.



Multi-Cluster Computing System

Background: Multiple-clusters

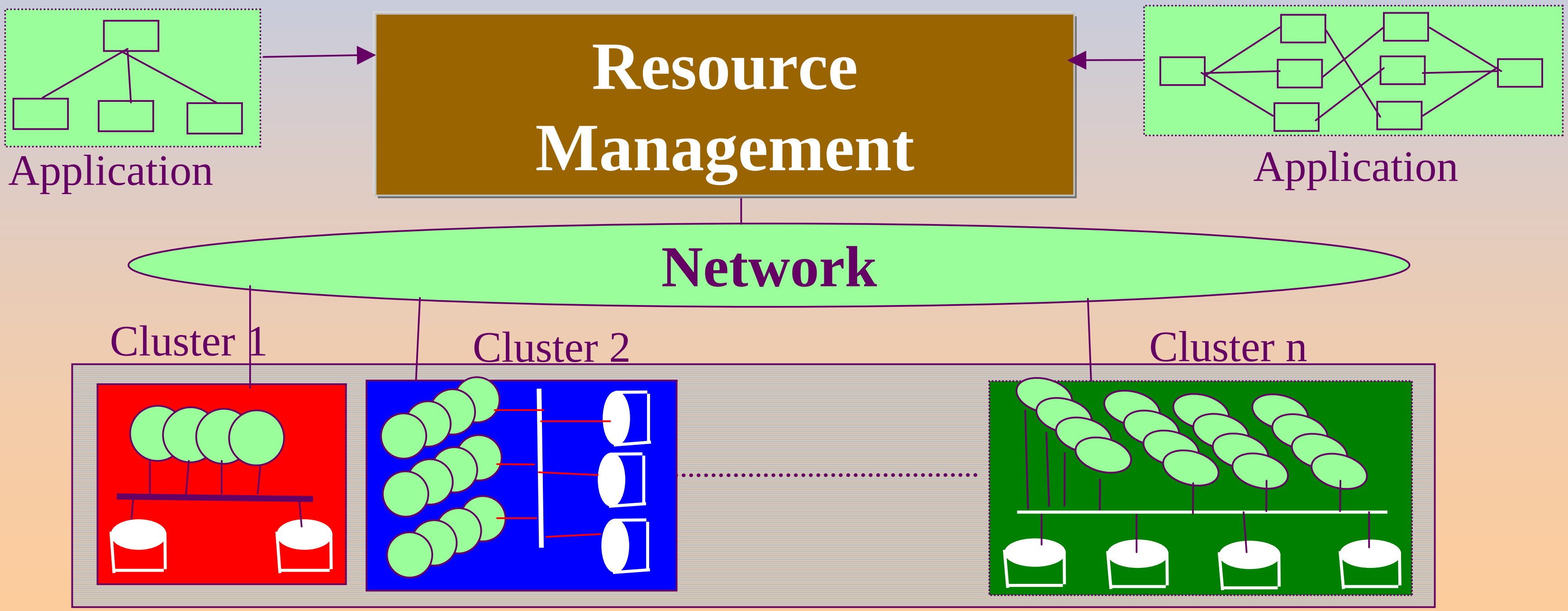
- **Multi-cluster Computing System is formed by interconnecting multiple clusters via WAN.**
- **Each cluster can have different number of workstations**



Multi-Cluster Computing System

Background: Applications

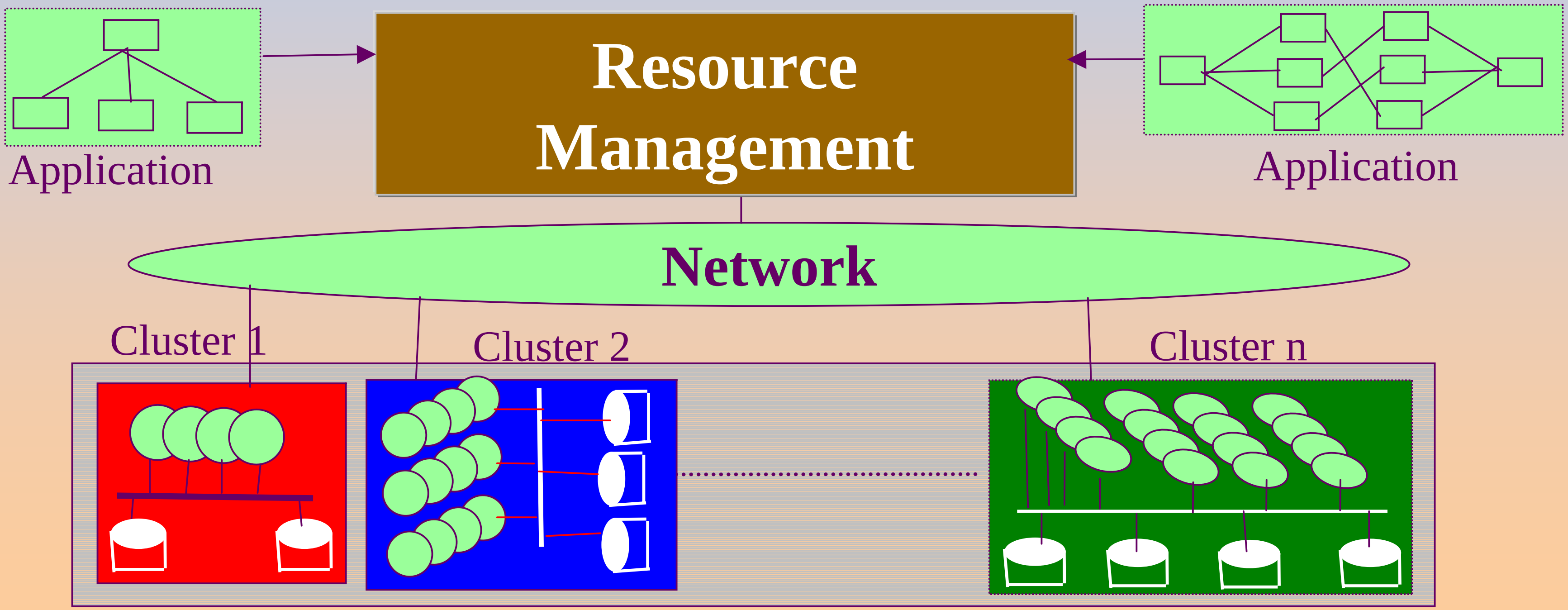
- Applications that can benefit from multi-cluster systems include:
  - parameter studies
  - probabilistic analysis



Multi-cluster computing system

Background: Resource Management

- **Proper resource management is a critical issue for multi-cluster computing systems:**
  - ☞ **Competing users**
  - ☞ **Diverse set of workloads coexist**
  - ☞ **Distributed nature of the resources**



Multi-cluster computing system

Background: Resource Management

- Current Focus is
  - On single-cluster system
  - Rarely address challenges posed by CC systems.
    - Scale of the system
    - Dynamically changing resource availability
    - Resource and workload heterogeneity.
    - Susceptibility to Failure
- To address these challenges we have to dynamically
  - Allocate resources to competing jobs
  - Re-allocate resources based on system states
- We developed and experimentally validated
  - Adaptive and scalable job scheduling policy
  - Manage multiple job streams across multiple clusters
  - Good response-time and system utilization

Problem Statement

- <u>Given</u>: A set of computationally intensive jobs, $J = \{J_1, \ldots, J_r\}$, that arrive stochastically
- A set of clusters, $S = \{C_1, \ldots, C_n\}$, each cluster with
  - $P = \{P_1, \ldots, P_m\}$ community-based workstations
  - $D = \{D_1, \ldots, D_k\}$ sharable disks.
- <u>Problem</u>: Schedule the J jobs onto the S clusters with objectives of good
  - Mean response time, and
  - System utilization
- <u>Constraints</u>:
  - No advance knowledge of resource availability, arrival and service times of the jobs/tasks.
  - A task can execute only after all input data received
  - At most one task can access any data storage at any given time.

Multi-cluster Computing Infrastructure

Scheduling Policy

- All incoming jobs are submitted to the system scheduler
- The scheduler is composed of three core components
  - Self-Scheduling
  - Pull Algorithm
  - Push Algorithm

Self-Scheduling

- We associate with each node in the cluster tree a parameter called base load level.
- Whenever the current load level < base load level:
  - ☞Request for a set of computation from parent
  - ☞A request that cannot be satisfied by a parent is backlogged and processed when jobs become available
  - ☞A request may recursively ascend the cluster tree
- Note that a node can have only one pending request at any given time

Pull Algorithm

- It is a form of space-sharing policy
- It assigns a partition size to jobs based on
  - ☞ Parent-child relationship; and
  - ☞ Negotiations between the nodes in the cluster.
- When it is invoked, the algorithm performs the following steps:
  - ☞ Determines an ideal number of jobs that can be assigned to a child
  - ☞ Adjust the number of jobs transferred by taking the request from child into account
  - ☞ Dispatch the jobs to the child

Push Algorithm

- Intrinsically allows jobs/tasks to be relocated
  - **from overloaded clusters to under loaded clusters**
- When the push algorithm is invoked, it performs the following steps
  - **First it determines the average load in the entire system**
  - **It then identifies those clusters that are above or below the average workload**
  - **Instruct over-loaded clusters to send a set of jobs to under-loaded clusters.**

Performance Evaluation

- **We used discrete event simulation to study the performance of the proposed policy and compare it with two baseline policies:**
  - **A derivative of the job-based time-sharing policy**
  - **An adaptive space-sharing policy originally introduced in Rosti et. al. and subsequently modified in Thanalapati and Dandamudi**
- Metrics used
  - **Mean response time - the sum of all job response time divided by the number of completed jobs**
  - **Average utilization - the job arrival rate times the mean service demand of the jobs divided by the number of processors in the system.**

Performance Evaluation

- **We used** synthetic workload with different characteristics:
  - **W1 – Matrix multiplication**
  - **W2 – Parameter Sweep**
  - **W3 – Burns Hat**
- System Environment
  - **Shared Homogeneous Environment: all processors run at the same speed, but workstations are shared by parallel and local workloads.**
  - **Dedicated Heterogeneous Environment: No background workload, but processors with different speed.**
  - **Shared Heterogeneous Environment: processors differ in speed and workstations may receive background workload at any time.**

Simulation Results and Discussions

Simulation Results and Discussions

Simulation Results and Discussions

Thank You ...