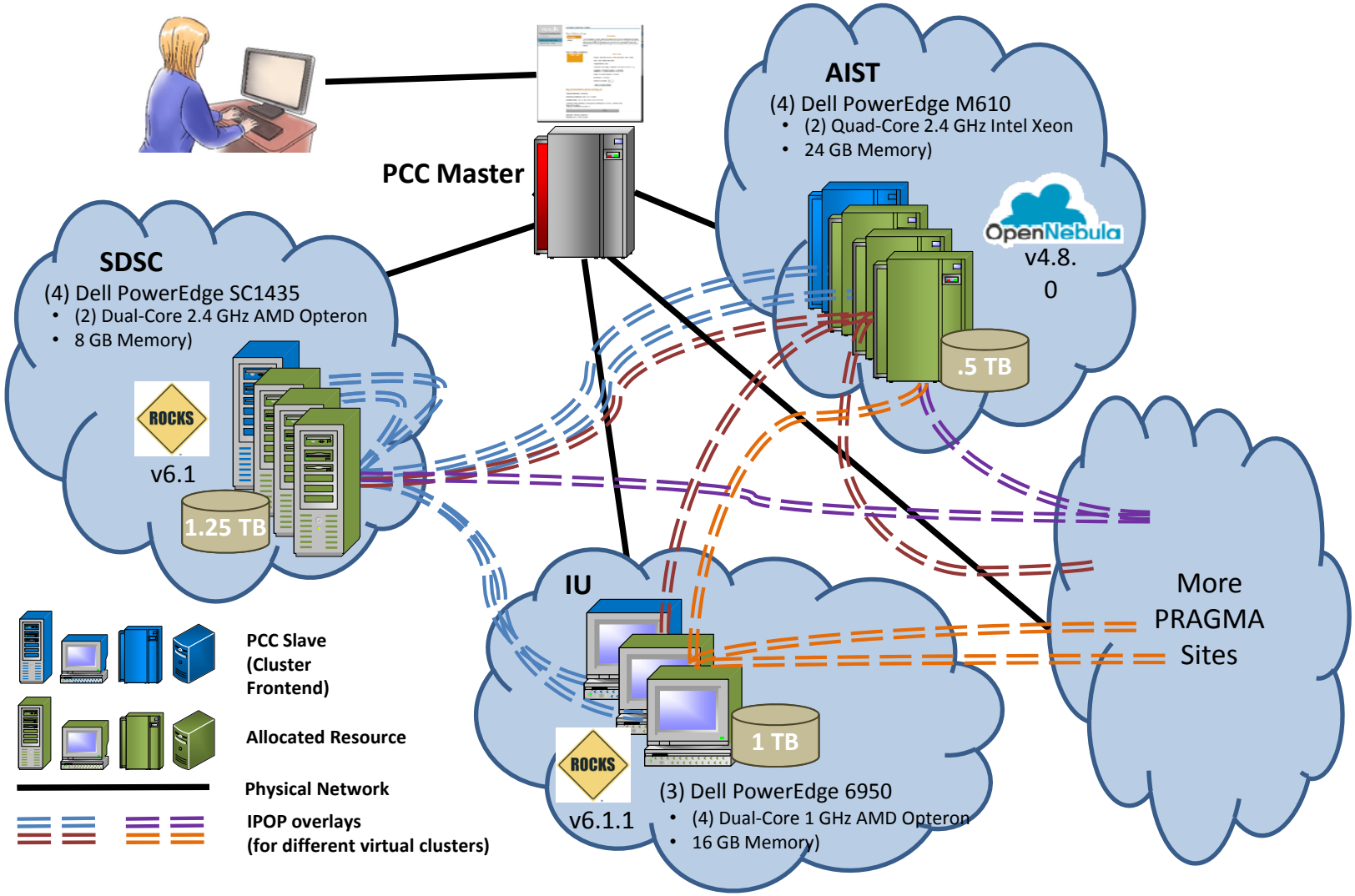
Distributed Cluster Management Architecture for Geographically Distributed Data Processing

Yuan Luo

School of Informatics and Computing, Indiana University Bloomington

PRAGMA 27 Students Workshop, Bloomington, IN, October 15th 2014

PCC Enabled PRAGMA Cloud



Problems Addressed

- User Controllability
  - Eliminate the gap between managing resources by resource providers and utilizing resources by applications
- Cloud Interoperability
  - If applications requires a larger number of resources, single-cloud solutions grow increasingly inadequate.
- Resource-Application Information Sharing
  - Information needs to be shared bi-directionally between resource providers and applications.

Contributions

- A resource allocation model to enable bi-directional resource-application information sharing
- A personal cluster controller that extends HTCondor and leverages IPOP to enhance user controllability.

Resource Allocation Model

Resource Specification

- **Cluster configuration**

  1) number of nodes per cluster; 2) CPU speed; 3) number of cores per node; 4) memory size; 5) disk size; 6) operating system; 7) hypervisor; 8) cloud platform; and 9) inner-cloud network bandwidth.

- **Inter-cluster network bandwidth matrix**

  The matrix contains end-to-end network bandwidth performance values between the frontend nodes of every two clusters.

- **Availability of each cluster**

  The remaining capacity of the cluster that a user can provision.

Application Specification

- **Resource requirements**

  1) number of nodes; 2) CPU speed; 3) number of cores per node; 4) memory size; 5) disk size; 6) hypervisor; 7) minimum network bandwidth.

- **Data locality**

  Data locality is described by an array of cluster names or IPs of data sources.

- **Application constraints**

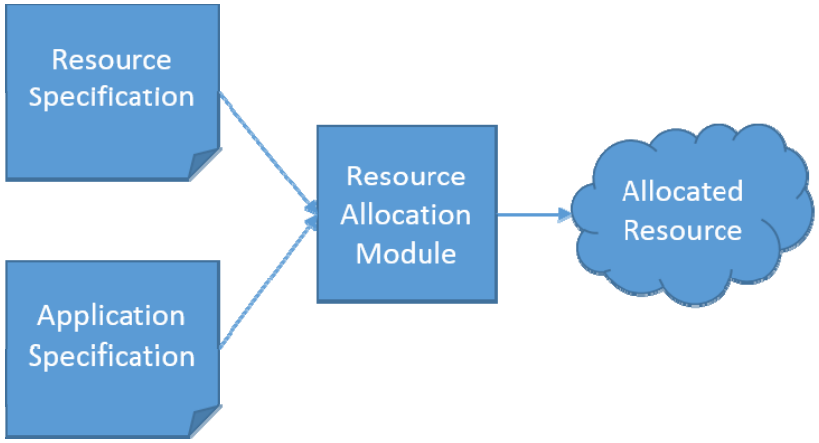
  The constraints are instantiated by the following three lists: a mandatory resource list, a forbidden resource list, and a preferred resource list.

- **Application characteristics:**

$$\alpha = (\alpha_1, \alpha_2, \alpha_3, ...) \begin{cases} 0 \leq \alpha_1 \leq 1, & \text{Compute Intensitive} \\ 0 \leq \alpha_2 \leq 1, & \text{Data Intensitive} \\ 0 \leq \alpha_3 \leq 1, & \text{I/O Intensitive} \end{cases}$$

Resource Allocation Model – *cont'd*

- Application to Resource Information Sharing
- Pluggable Resource Allocation Algorithms (sample)



Algorithm 1 Resource Allocation Algorithm

Require: Resource Specification $RS[]$, Application Specification AS

1: **procedure** RESOURCE-ALLOCATION($RS[], AS$)
2: $CandidateList$=GETCANDIDATELIST($RS[], AS$);
3: **if** $\alpha < 1$ **then**
4: $CandidateList$=SORTBYCP($CandidateList$);
5: **else if** $\alpha > 1$ **then**
6: $CandidateList$=$AS.datalist[]$;
7: $CandidateList$=APPENDLIST($CandidateList$, $RS.matrix[][]$);
8: **end if**
9: ALLOCATIONPLAN($CandidateList$,$AS.req[]$);
10: **end procedure**

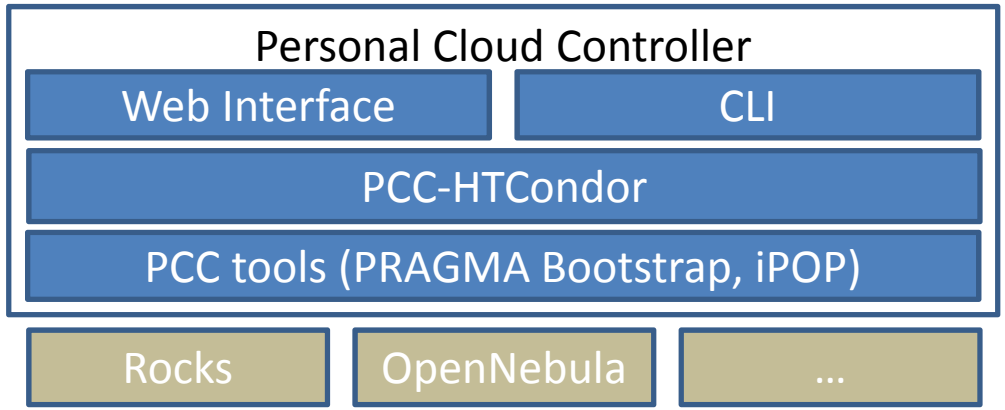
Personal Cloud Controller (PCC)

- PCC creates virtual clusters by allocating virtual machines from multiple cloud platforms, builds a network overlay, and manages the virtual cluster life-cycle in a fault-tolerant manner.

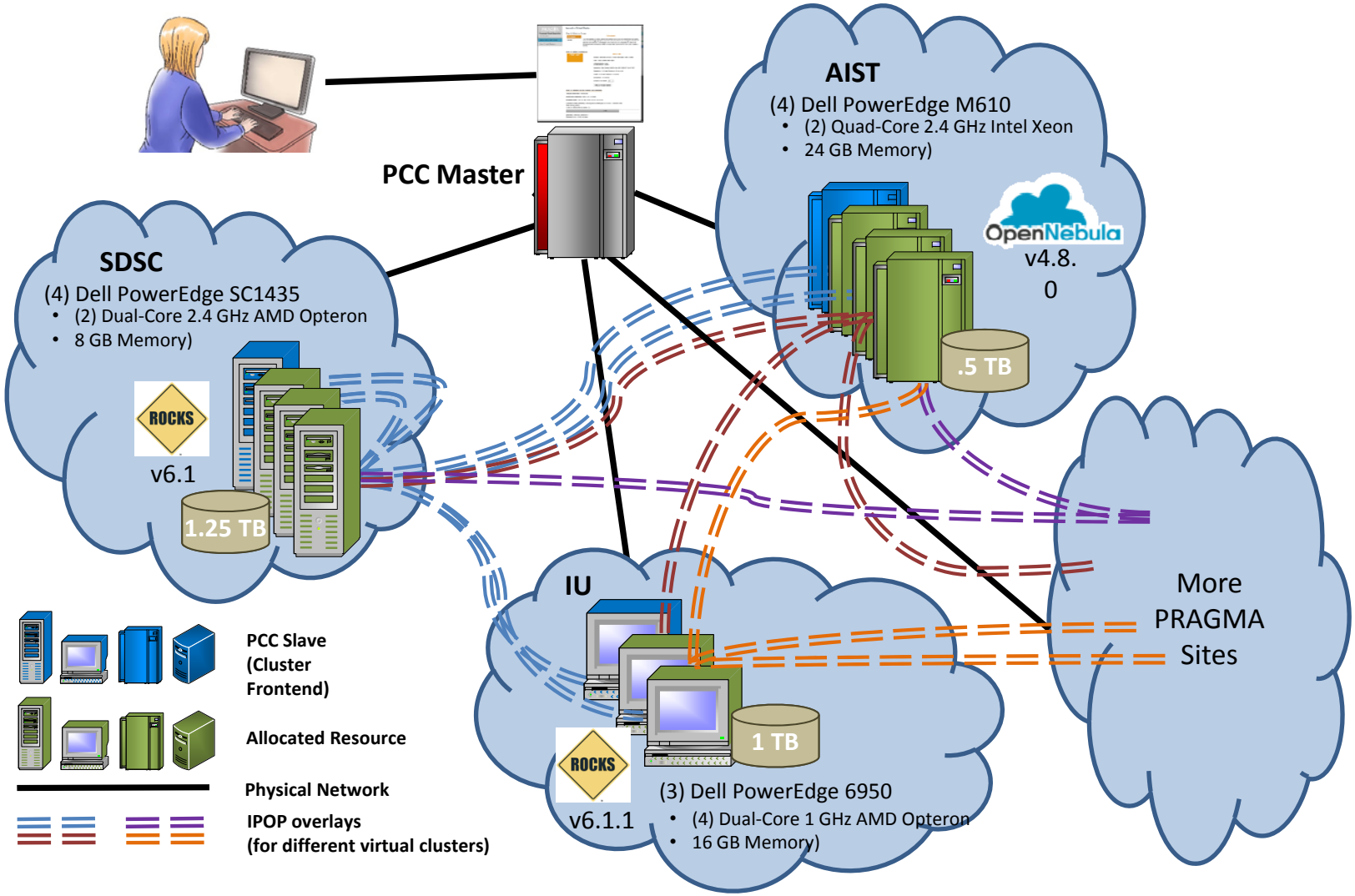
PCC Modules	Description	Implementation
Resource Registry (RR)	A cache that contains resource specifications (RS), frequently updated to latest status from cloud providers.	Customized HTCondor ClassAds and cron job scripts.
Resource Allocator (RA)	Provisions resources utilizing the resource allocation module described in the previous section.	HTCondor MatchMaking procedures
Allocation Manager (AM)	Starts, stops, monitors, and handles fault tolerance of the allocated resources.	PCC-HTCondor (DAGMan, PRAGMA Bootstrap)

Personal Cloud Controller *– cont'd*

- Personal Cloud Controller (PCC) includes an extended version of HTCondor, called PCC-HTCondor that
  - ✓ Provides flexible interface to enable a high level of user controllability;
  - ✓ Automates configuration, deployment, and fault recovery;
  - ✓ Parallel virtual cluster deployment using HTCondor DAGMan.
- Personal Cloud Controller (PCC) integrates PRAGMA tools to easily deploy and manage virtual clusters.
  - ✓ PCC integrates PRAGMA Bootstrap to instantiate virtual clusters.
  - ✓ PCC integrates IPOP via PRAGMA Bootstrap to build network overlays.



PCC Enabled PRAGMA Cloud *- revisit*



PCC Evaluation

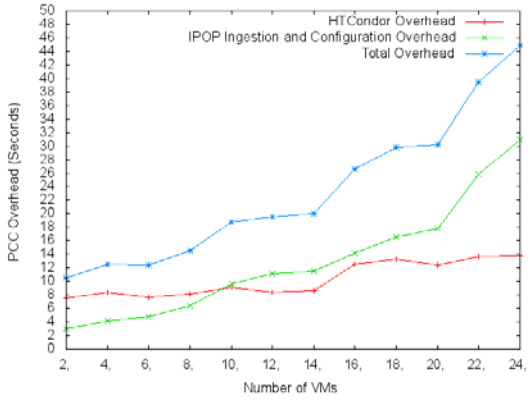
- We measure overhead of PCC as captured by overhead during the resource provisioning phase and overhead of application running over VPN.
- Testbed. Two clusters were selected: one at Indiana University(IU) and the other at the San Diego Supercomputer Center (SDSC).

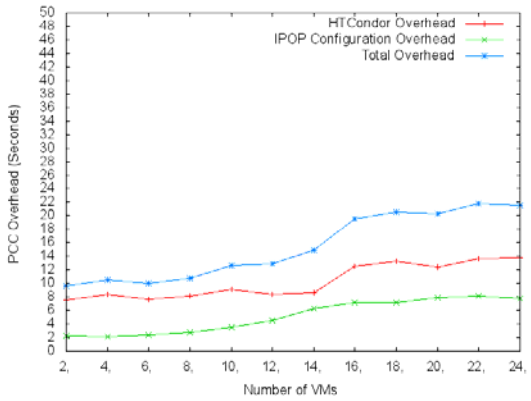
Table 1. Testbed Specifications

Cluster	Nodes	CPU	Cores	Mem	Ethernet	OS	VMM	Cloud Platform
SDSC	4	2.4GHZ	4	8GB	1000Base-T	CentOS 6	KVM	Rocks 6.1
IU	3	2.4GHZ	8	16GB	1000Base-T	CentOS 6	KVM	Rocks 6.1

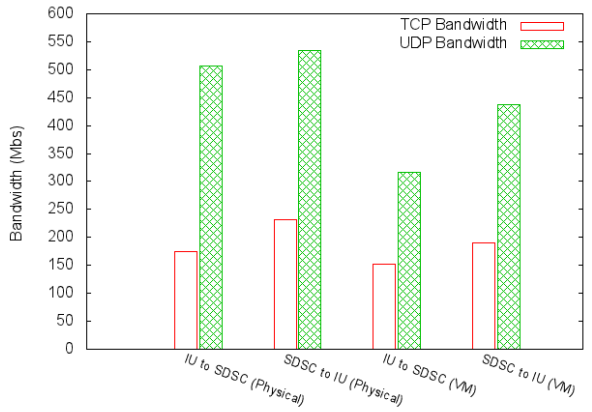
PCC Evaluation – *cont'd*

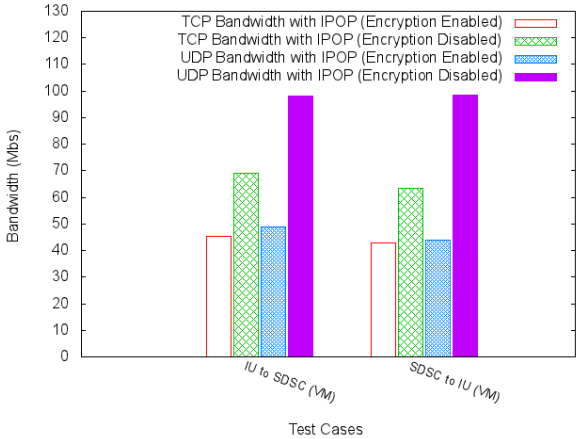
PCC Overhead Evaluation

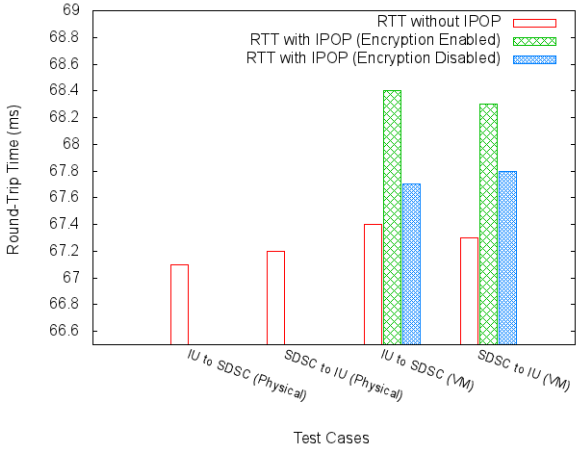




Network Overhead Evaluation







Ongoing and Future Work

- Perform a comprehensive evaluation against PCC.
- Improve resource allocation algorithms.
- Enable resource to application information sharing.
- Extend the Hierarchical MapReduce model to support distributed sensitive data processing.
- Schedule application jobs based on VC topologies, and VM provenance.

Acknowledgement

- The author would like to thank
  - Dr. Beth Plale (Indiana University), Dr. Philip Papadopoulos, and Shava Smallen (UCSD)
  - Luca Clementi (SDSC) and Pongsakorn U-chupala (NAIST) (initial authors of PRAGMA Bootstrap)
  - Dr. Renato Figueiredo and Dr. Pierre St Juste (UF) (IPOP authors)
- This work funded in part by NSF Award OCI 1234983

Thank you!