

Virtual Clusters on the Fly

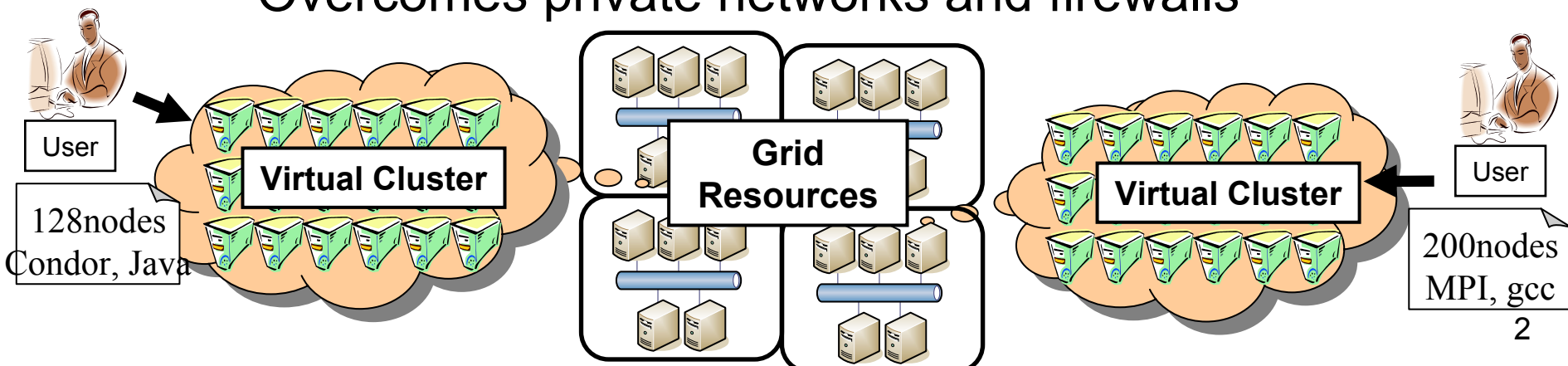
Fast, Scalable, Flexible Virtual Cluster Installation

Hideo Nishimura[†]
Naoya Maruyama[†]
Satoshi Matsuoka^{†‡}

[†]*Tokyo Institute of Technology*
[‡]*National Institute of Informatics*

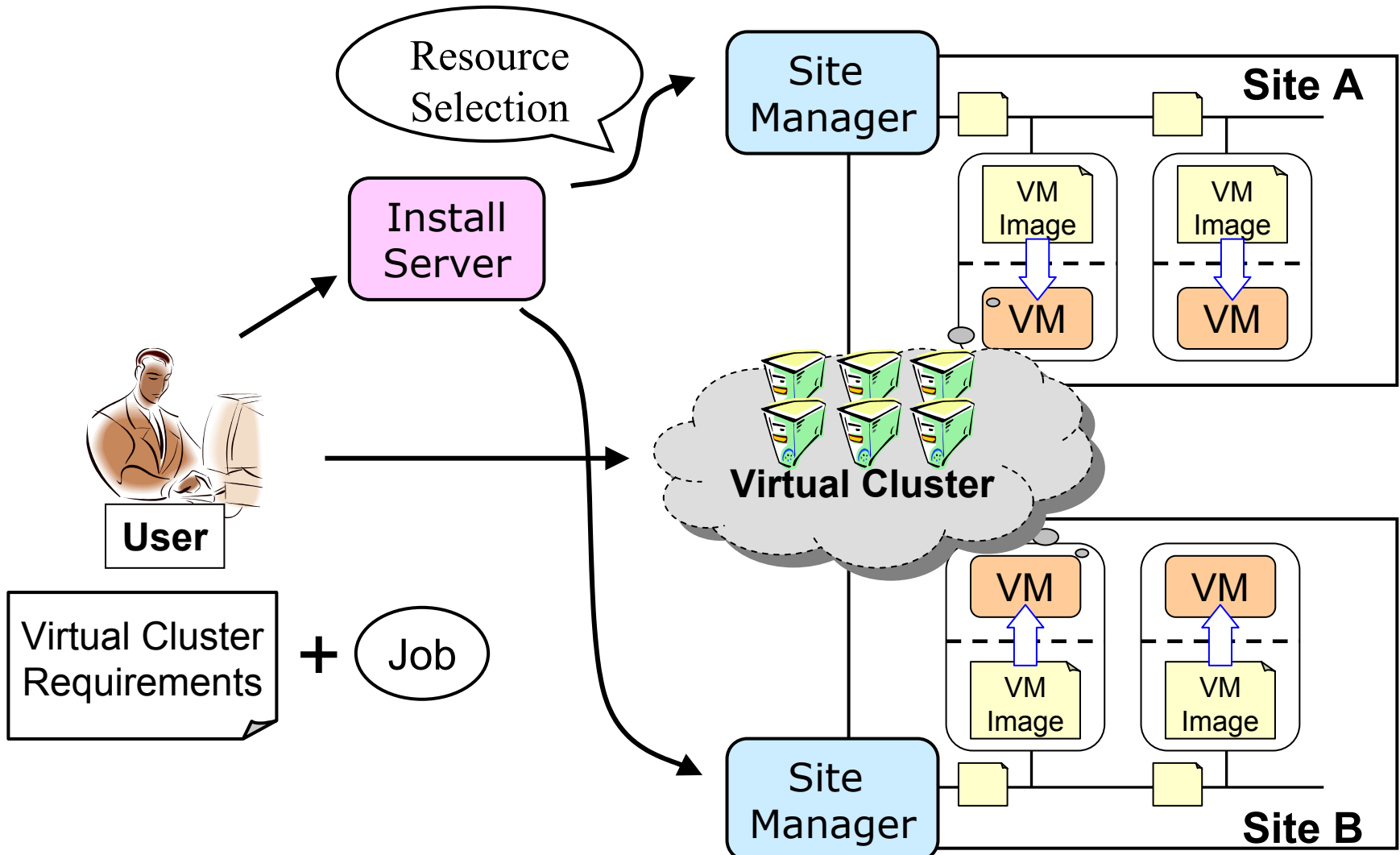
Grid Resource Sharing with Virtual Clusters

- *Virtual Cluster*
 - Virtual Machines (VM) as computing nodes
 - Customizable without interfering with underlying real environments
 - Hides software heterogeneity
 - Seamless integration with user's own resources
 - Interconnected via overlay networks
 - Hides network asymmetry
 - Overcomes private networks and firewalls



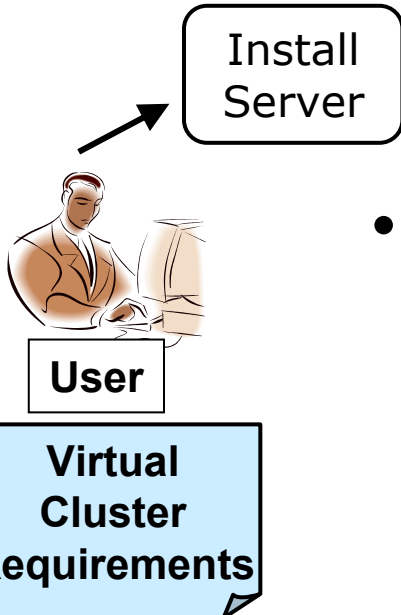
Deployment of Virtual Clusters

Dynamically deploy a VC for each user job



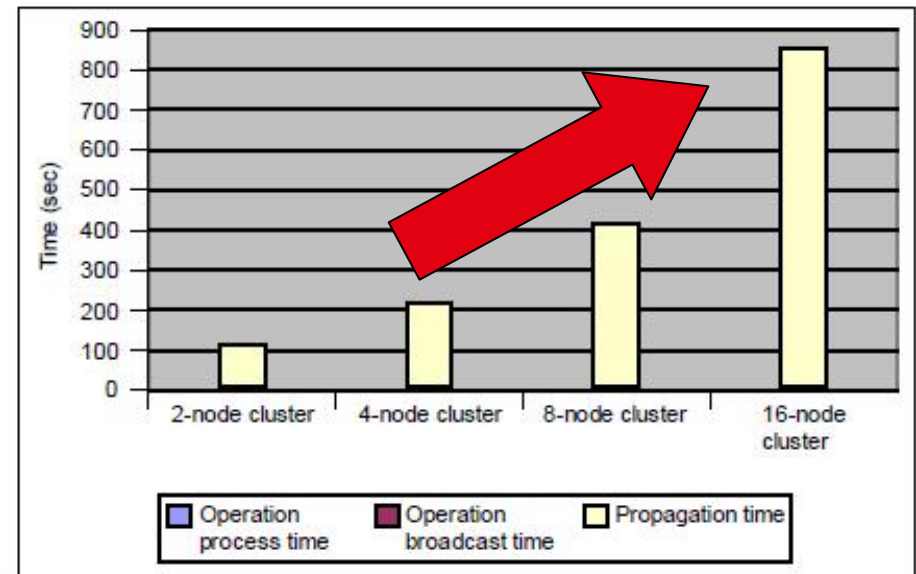
Challenge: Flexibility

- Requirements
 - Fully customizable
 - Easy for end users to specify
- Previous approaches [Krsul04, Keahey05]
 - User-created VM images
 - Non-trivial task for end users
 - Statically-prepared VM images by sysadmin
 - Impractical to predict all the possible desired VM configurations
- *Our approach*
 - *Automatically creates VM images by extending an existing cluster configuration tool*



Challenge: Scalability

- Requirements
 - Scalable installation with increasing number of VMs
- Previous approaches
 - Naïve method like NFS sharing [Foster06]
 - poor scalability
 - 15 min for 16 VMs
- *Our approach*
 - *Pipelined near- $O(1)$ data transfer*
 - *highly scalable*

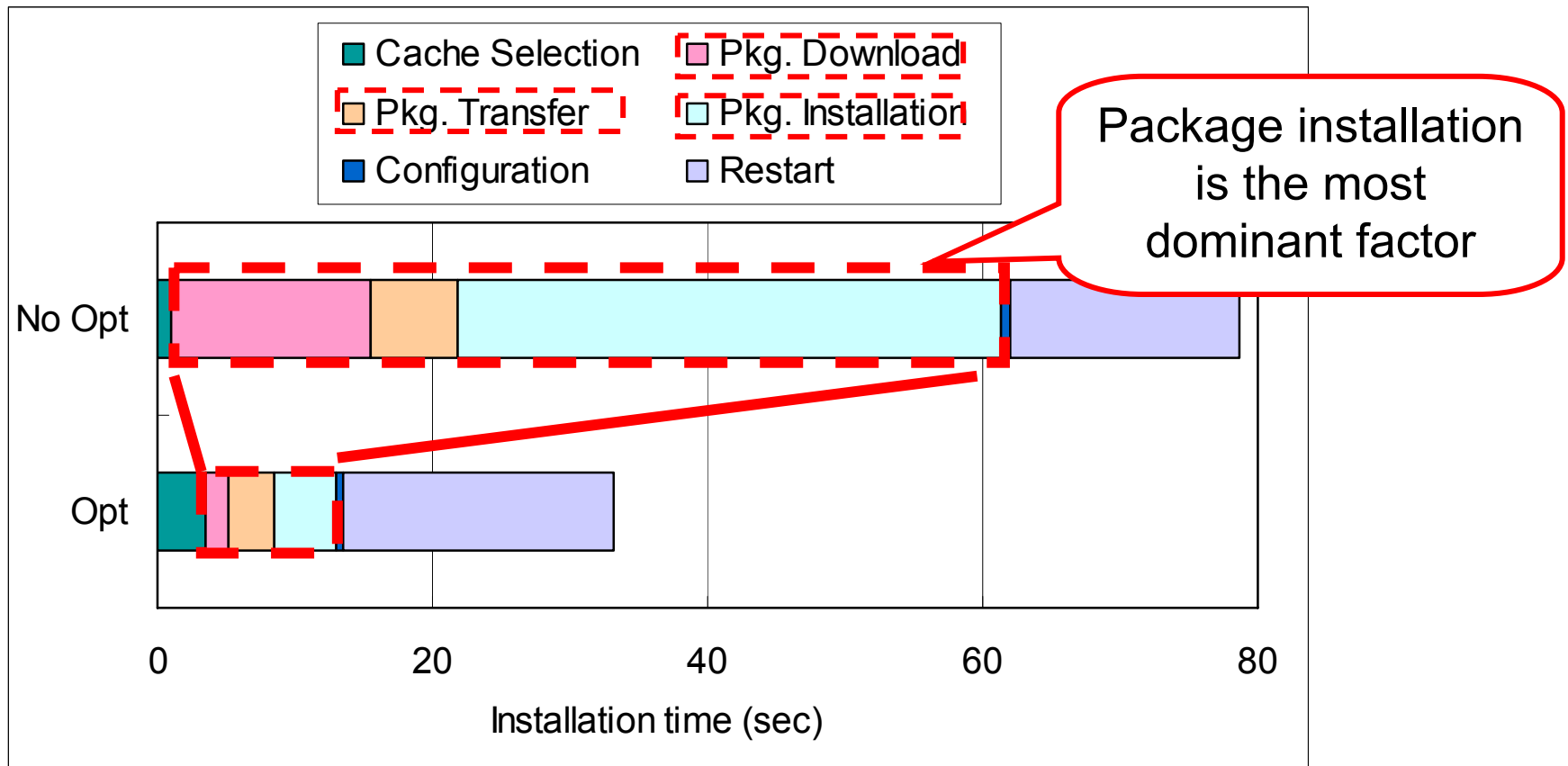


(Extracted from [Foster06])

Challenge: Instantaneity

- Requirements
 - Fast installation

- *Our approach*
 - *Dramatically reduces package installation time with virtual disk image caching*



Contributions

- A novel virtual cluster installation technique
 - **Flexible** customization using an existing cluster management tool (LUCIE, ROCKS support planned)
 - **Scalable** image transfer with a pipelined transfer technique and
 - **Fast** installation with *intelligent VD image caching (main focus of this work)*
- Demonstration of the proposed technique
 - Achieved installation of 190 VMs in as fast as 40 seconds.
 - Extrapolation suggests a 1000-VM cluster can be installed in less than 2 minutes.

Agenda

- ✓ 1. Introduction
- 2. Optimization Algorithm
- 3. Architecture and Implementation
- 4. Experimental Evaluations
- 5. Related Work
- 6. Conclusions

Optimization of Pkg. Installation Time

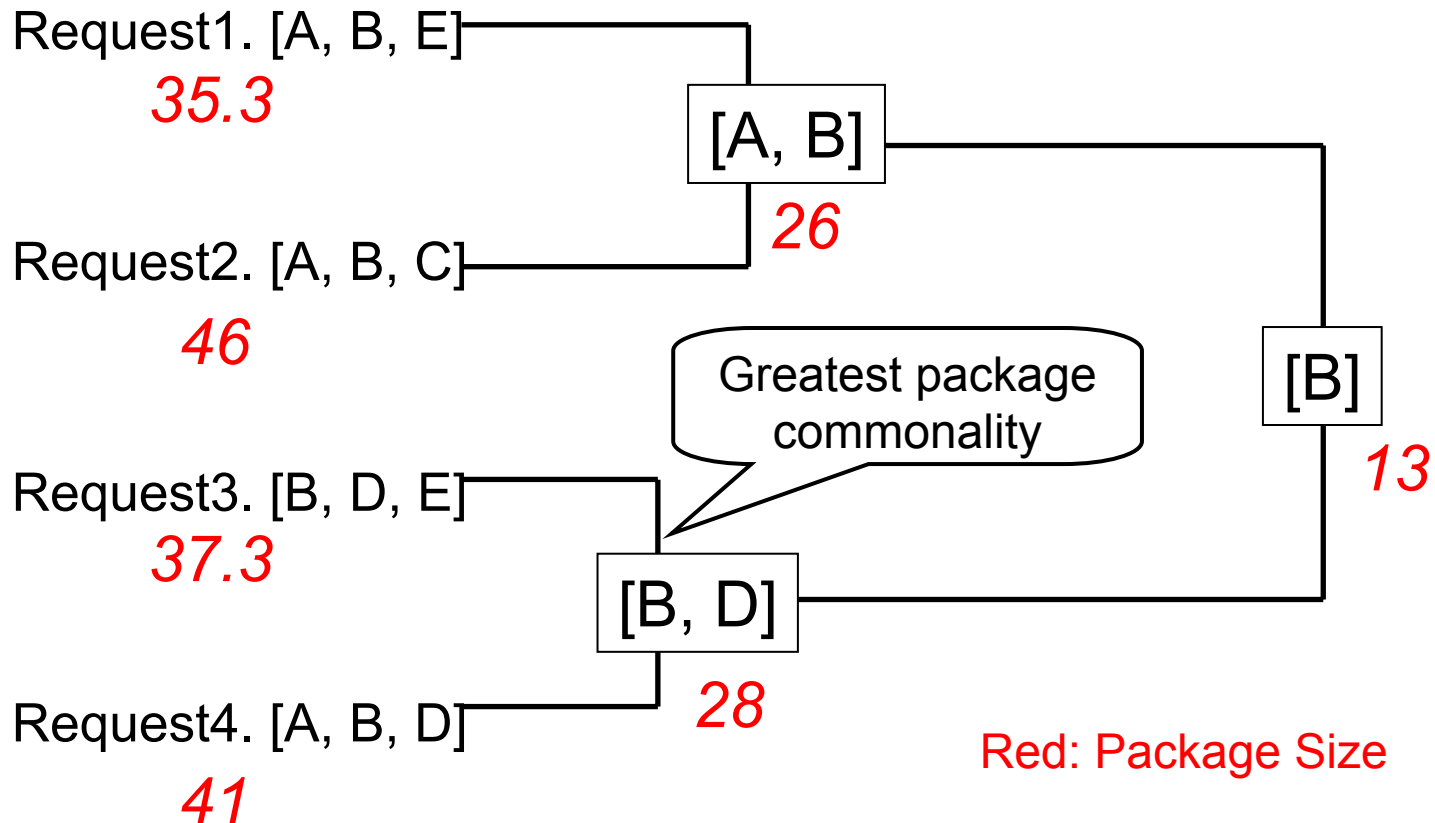
- Observation
 - Some packages appear more frequently (GCC, MPI, etc.)
- Basic Idea
 - Automatically identify combinations of frequently used packages
 - e.g., { [MPI make gcc], [condor blast], [MPI gcc], [condor java], ... }
 - { [MPI gcc], [condor], ... } (Don't select [condor blast], etc)
 - Create VM disk images including only those combinations (**VD Cache**)
 - For a new VM: install only the missing packages that are not included in the VD cache.
- *Q: How to determine what combination to include in the cache image?*

VD Caching Algorithm

- Hierarchical clustering of user-requested package sets
 - Dissimilarity between requests
 - *(the size of common packages) ⁻¹*
 - Effectively identifies commonality among packages
 - Works with installation trace history
 - *Automatic adaptation* to particular package selection tendencies

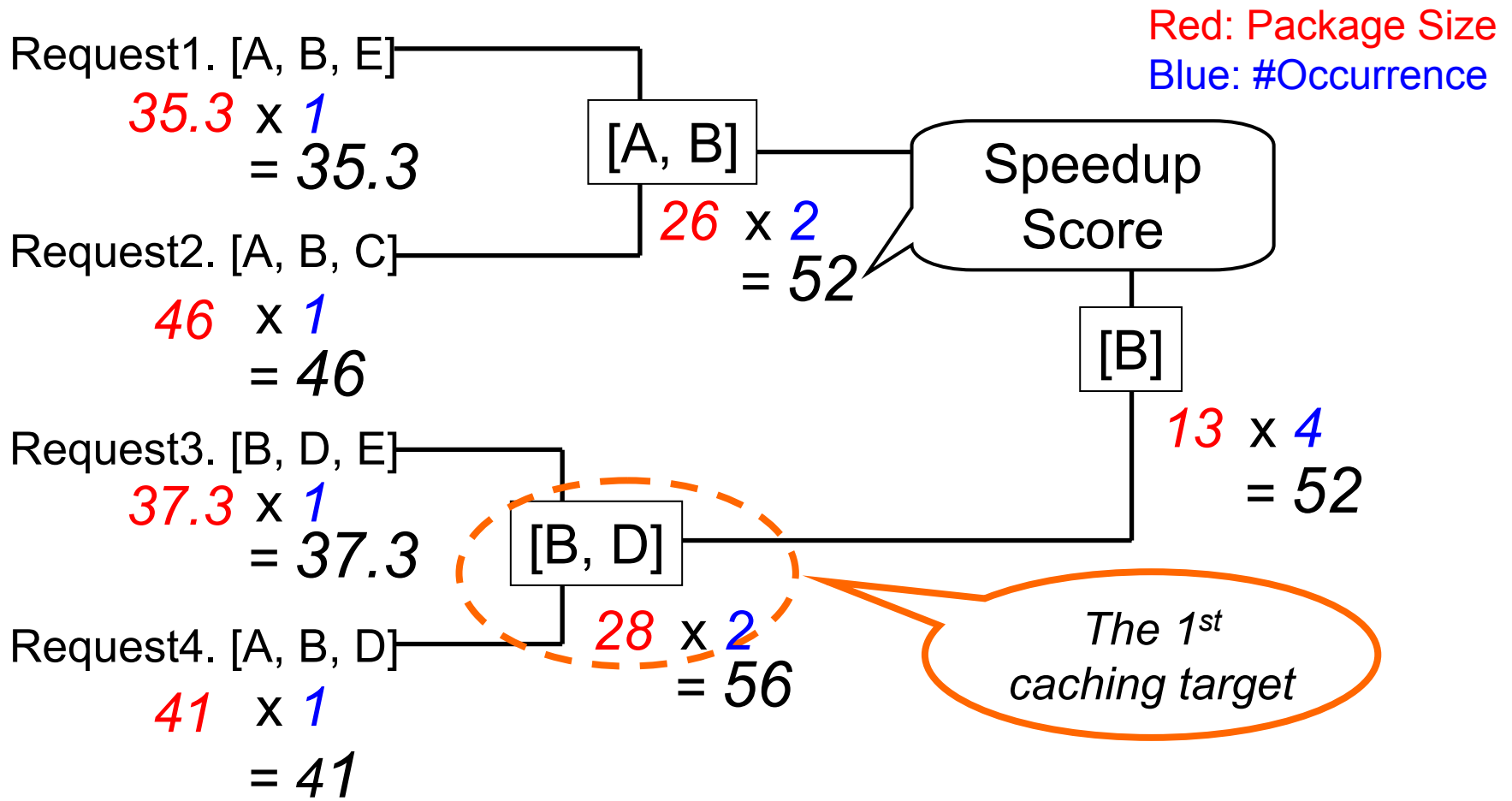
Clustering User Requests

Package	A	B	C	D	E
Size	13	13	20	15	9.3



Selecting the Clusters to Cache

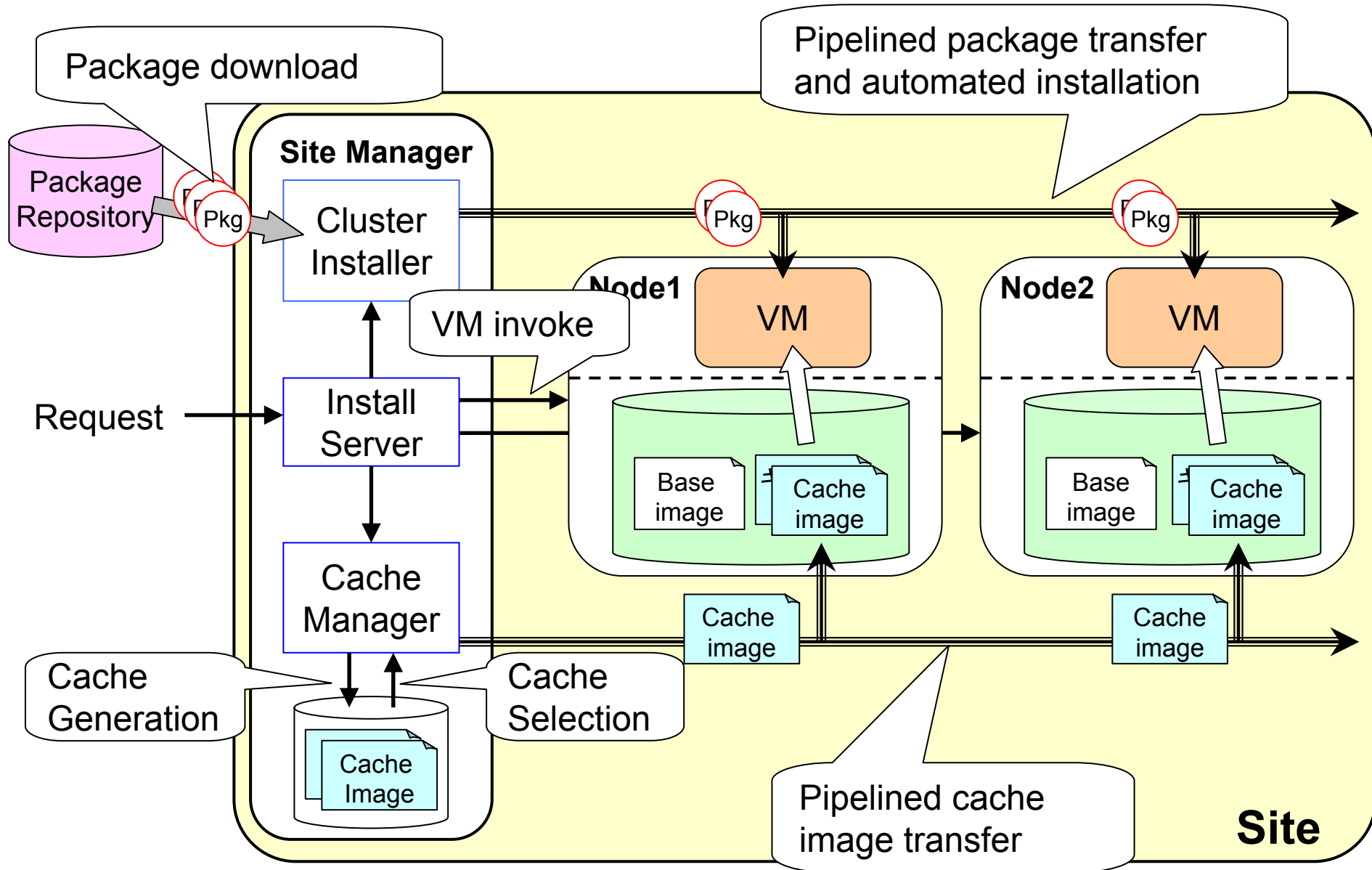
- *Speedup score* = sizeof(Packages) x (#Occurrence)
 - Reflects the expected time reduction if the cluster is cached.
- Select the clusters with the largest scores within a given disk space for VD caches.



Agenda

- ✓ 1. Introduction
- ✓ 2. Optimization Algorithm
- 3. Architecture and Implementation
- 4. Experimental Evaluations
- 5. Related Work
- 6. Conclusions

Overall Architecture



Overview of Prototype Implementation

- Xen as a Virtual Machine Monitor [Barham03]
- Lucie as the base cluster installer [Takamiya06]
 - Using Debian Linux and package management utility DPKG/APT
 - Other tools could be used as well (NPACI/SDSC ROCKS, Oscar, etc)
- Dolly+ [Manabe01] for pipelined transfer

Agenda

- ✓ 1. Introduction
- ✓ 2. Optimization Algorithm
- ✓ 3. Architecture and Implementation
- 4. Experimental Evaluations
- 5. Related Work
- 6. Conclusions

Experimental Evaluation

- Evaluation Metrics
 - Virtual cluster installation time reduction
 - Scalability with increasing number of VMs
 - Only single-site scenarios; multi-site evaluations planned
- Evaluation platform

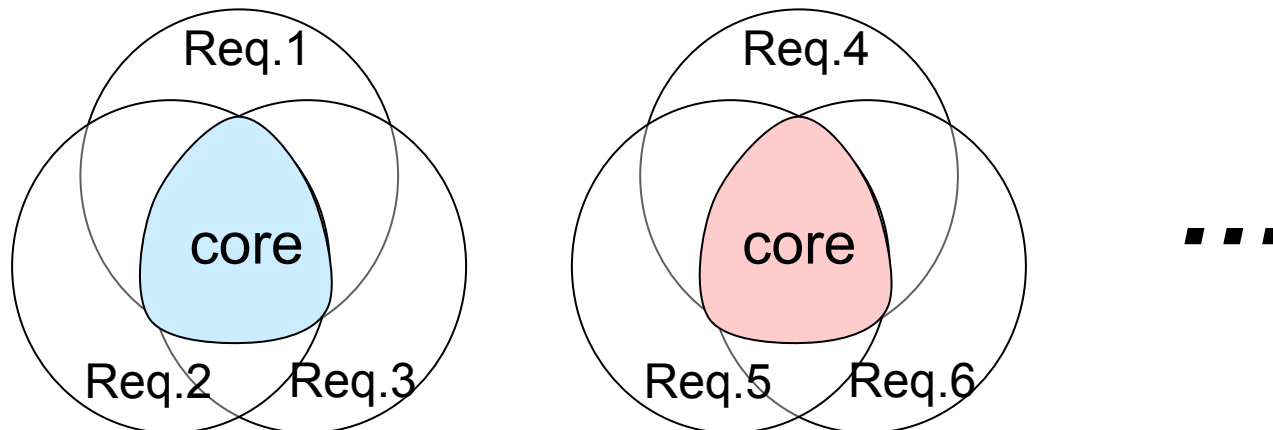
Software	Xen	Lucie	Dolly+	MPICH
Version	3.0.2-2	0.0.5	0.93-1	1.2.7p1

	CPU	RAM	HDD	Network	OS
Site Manager	Dual Athlon2000+	1GB	IDE	Gigabit Ethernet	Linux-2.6.12.6
Node Type 1	Dual Opteron242	2GB	IDE		Linux-2.6.16-xen
Node Type 2	Dual Opteron280	4GB	SATA		
Node Type 3	Dual Opteron250	2GB	SCSI		

Note the heterogeneity of CPU and HDD

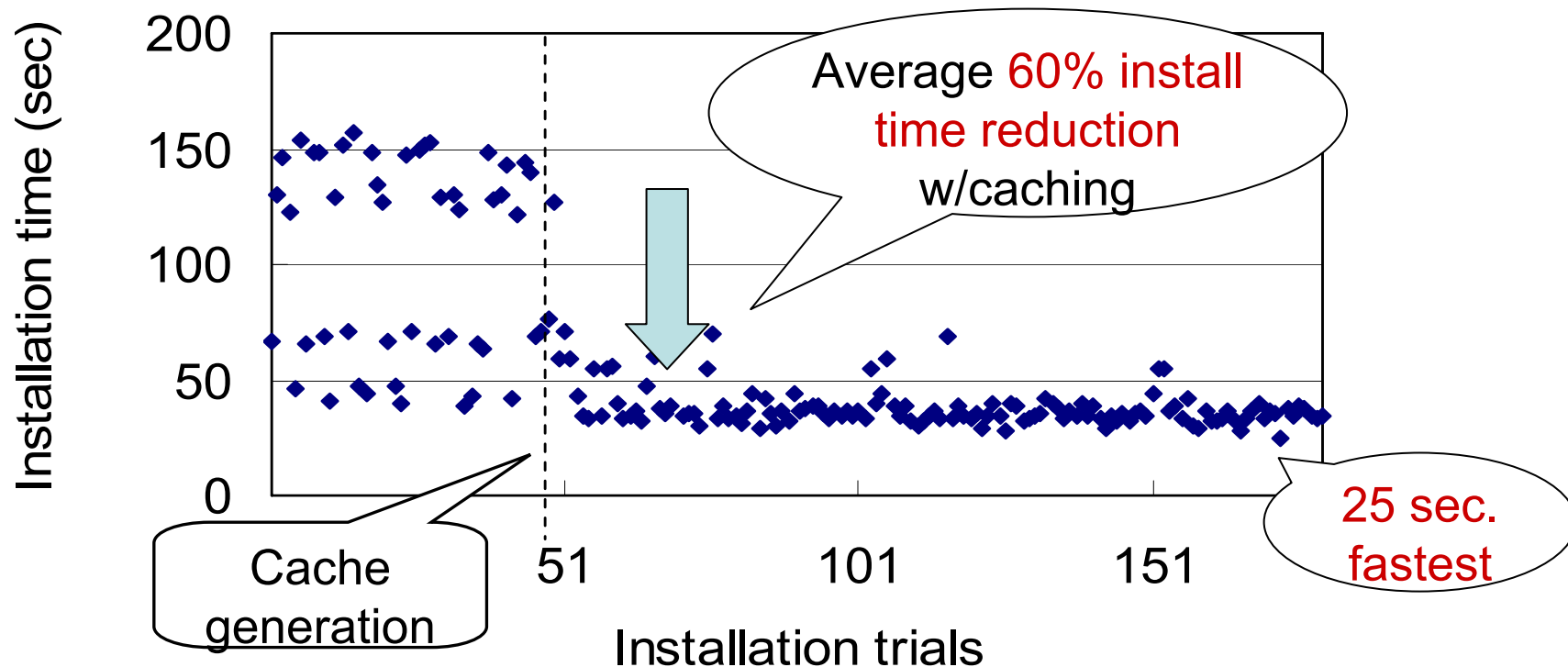
Artificial Request Generation

- Still difficult to collect large set of real requests for benchmarking => generate artificial requests
- Assume requests consist of some number of *core* sets with varying small *extra* packages
 - core: {Condor, Bio}, {Condor, Numeric}, {MPICH, Bio}, and {MPICH, Numeric}
 - extra: 45 distinctive package sets
- 180 (= 4 * 45) distinctive package requests generated

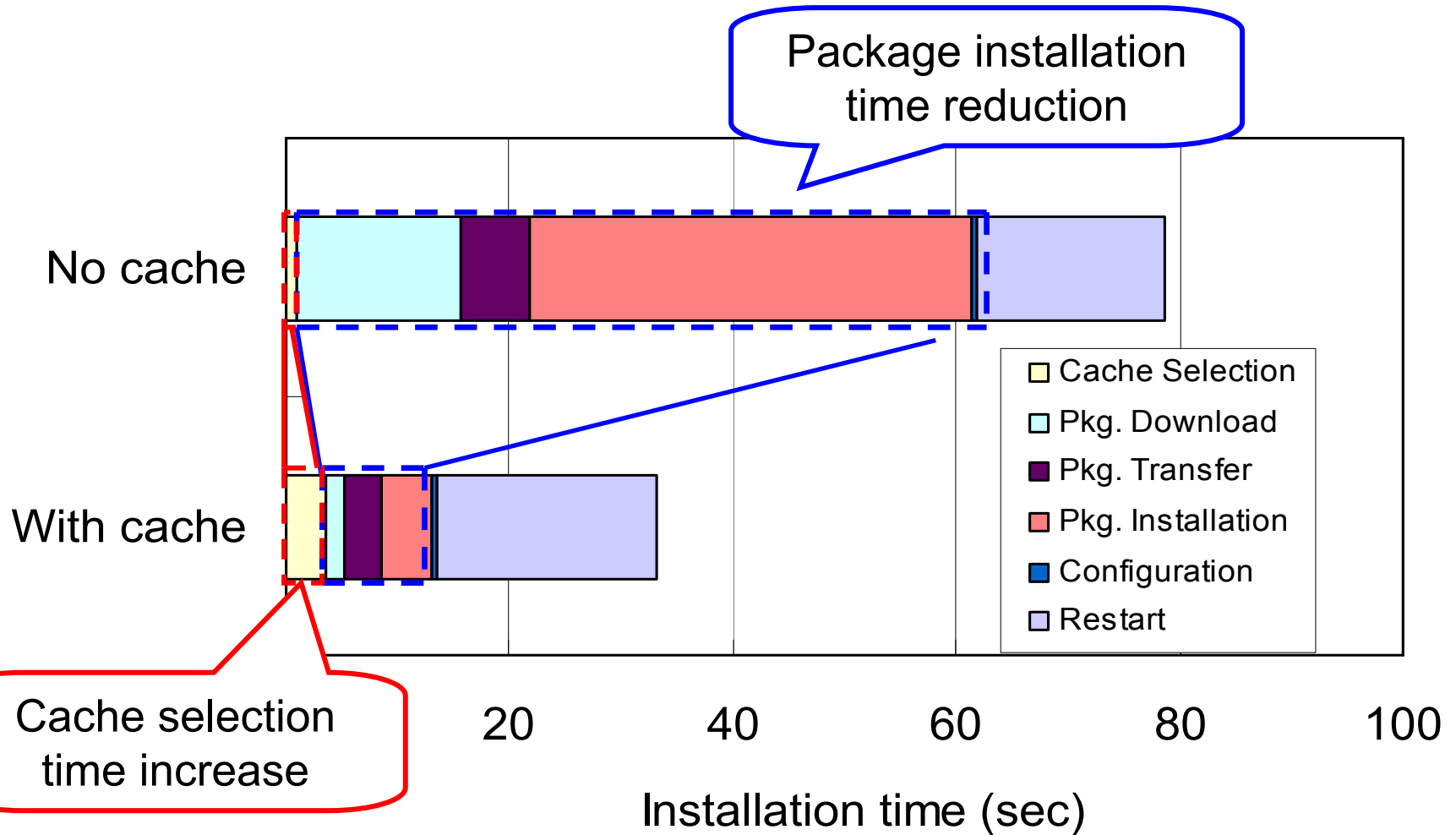


Effect on VPC Construction Time

- 200 installation requests for 67-VM virtual clusters
- Generate cache images every 50 installations within 5GB cache space
- 40 to 160 sec → 25 to 70 seconds by VD caching

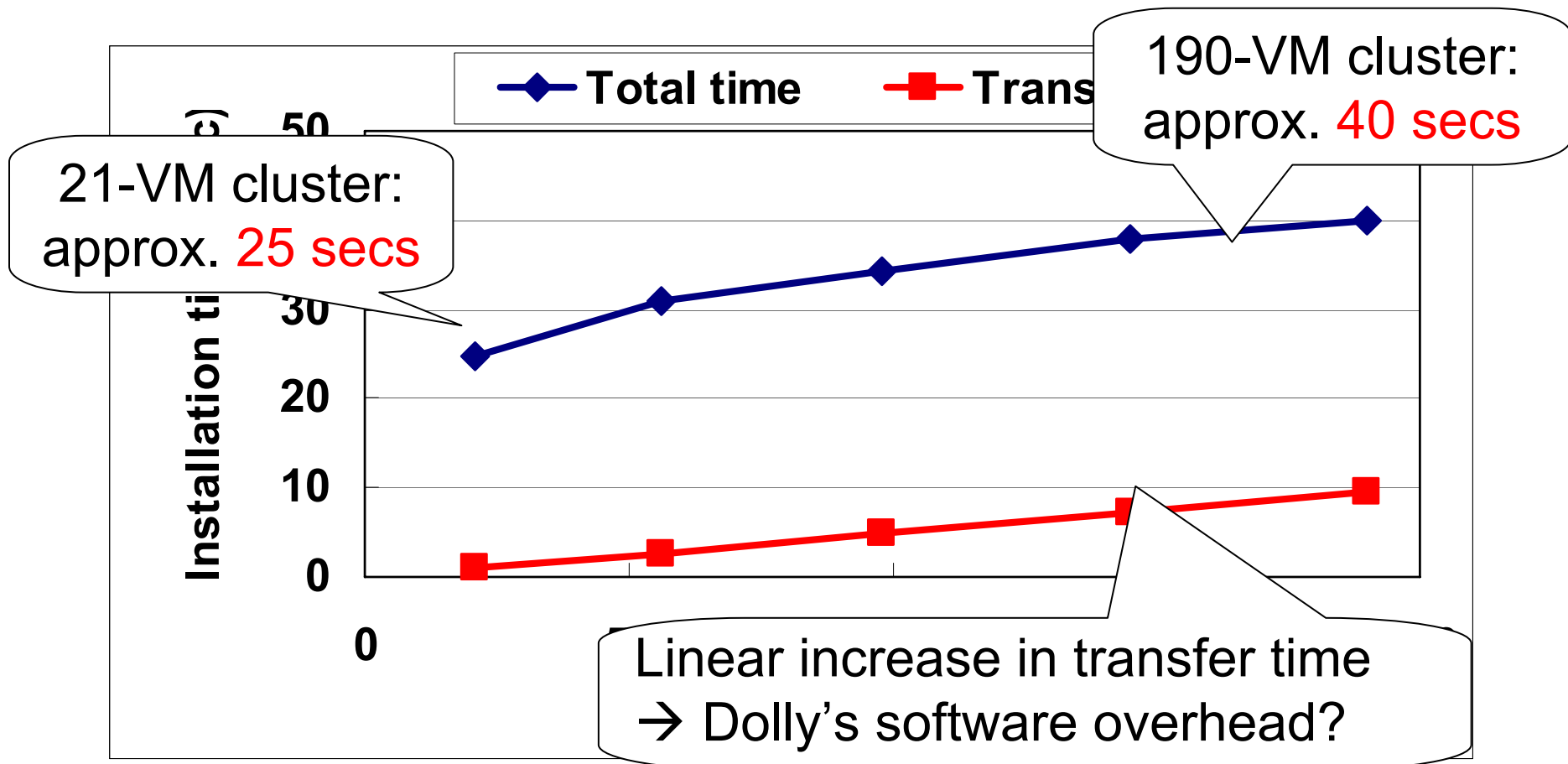


Breakdown of Installation Time

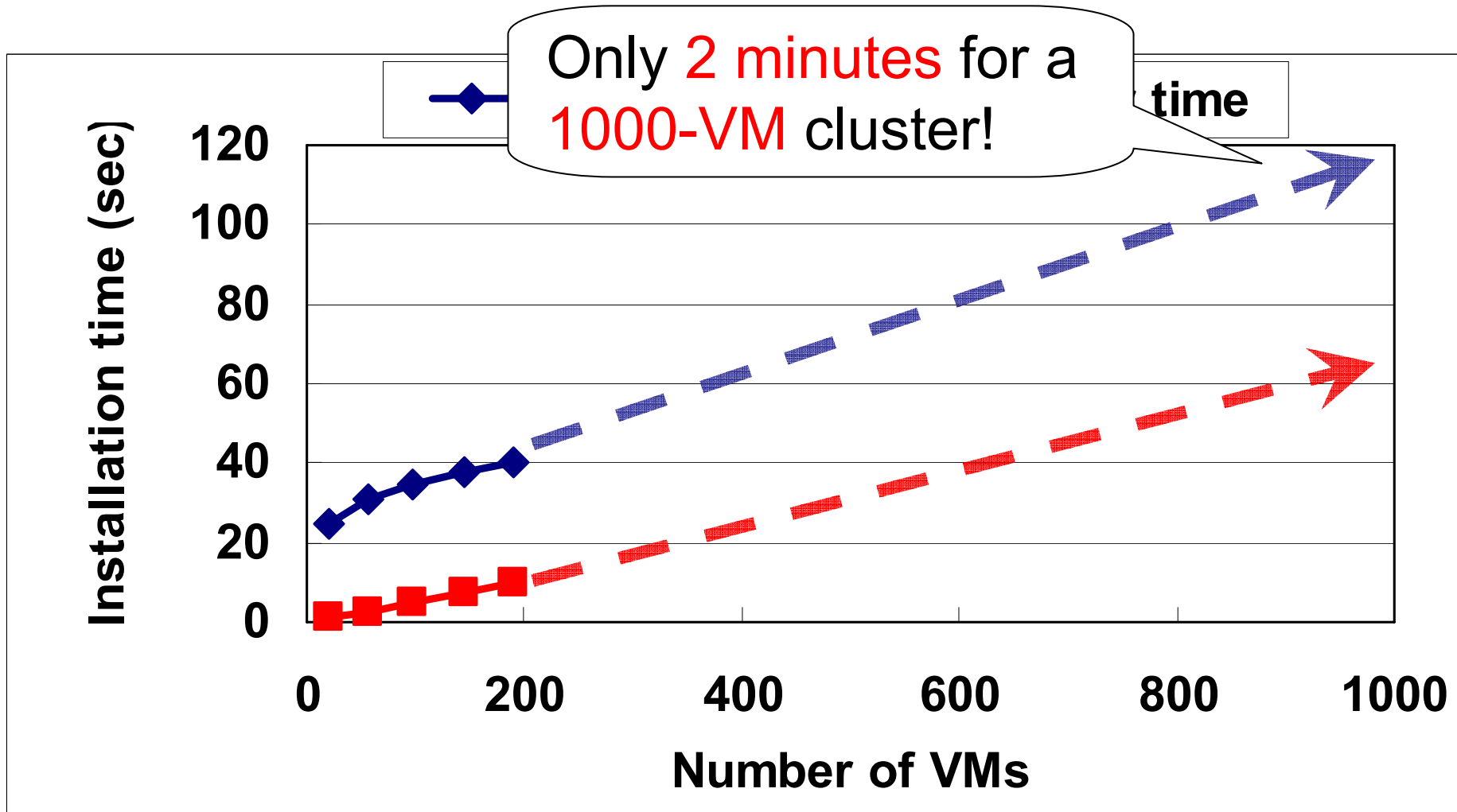


Scalability w/# of VM nodes

- Installation time of virtual clusters up to 190 VMs
 - Already-deployed VD cache + 5.8MB new packages

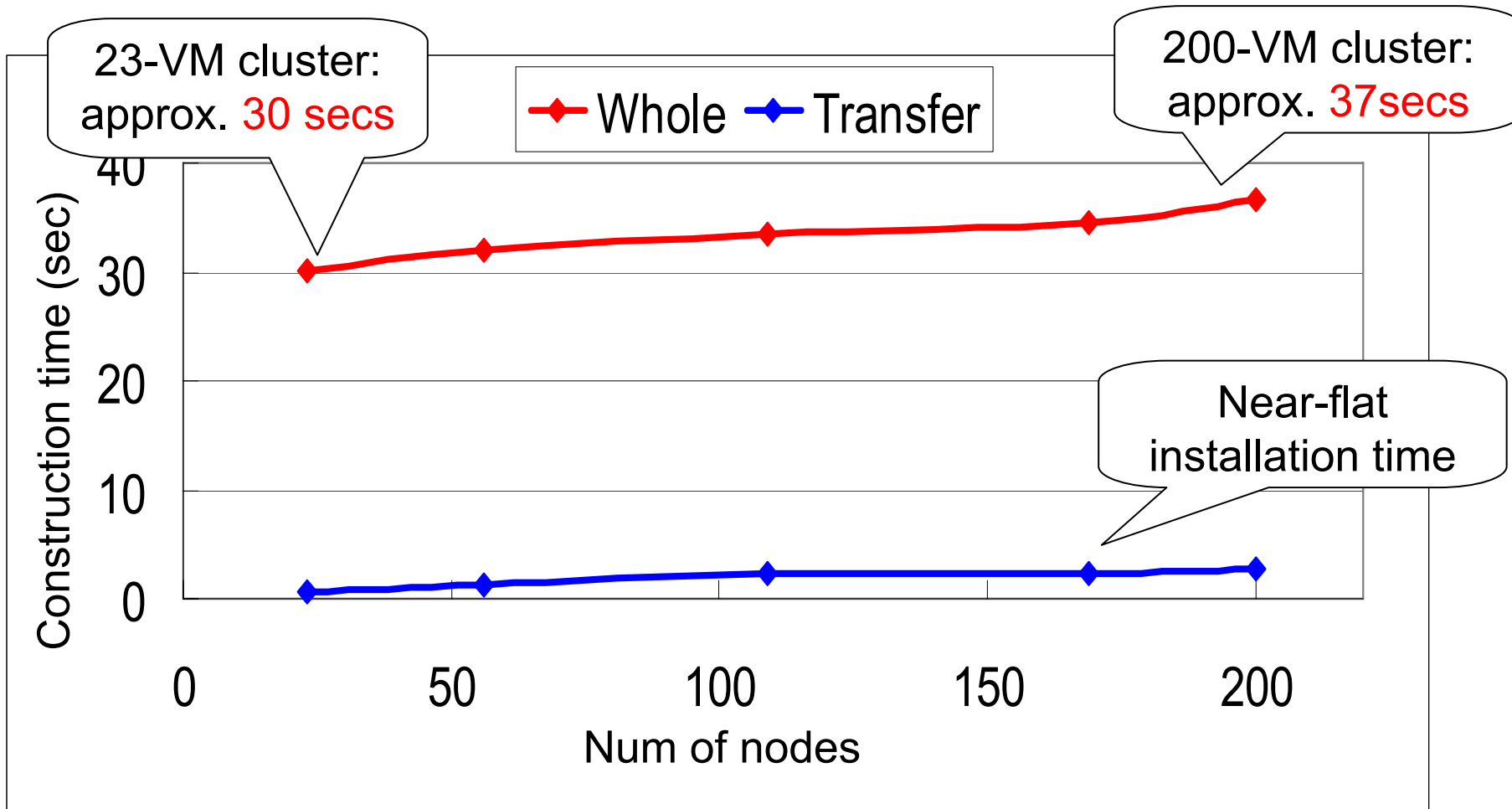


Optimistic Extrapolation to 1000 VMs



Recent Results

- Uses MPICH's bcast as a transfer substrate



Agenda

- ✓ 1. Introduction
- ✓ 2. Optimization Algorithm
- ✓ 3. Architecture and Implementation
- ✓ 4. Experimental Evaluations
- 5. Related Work
- 6. Conclusions

Related Work

- Integration of virtual clusters with existing Grid services
 - Virtual Workspaces [Keahey05]
 - Our method could be used as a fast installation engine
- Environment configuration
 - VMPlants [Krsul04]
 - DAG-based customization with statically-prepared VM images.
 - Amazon EC2
 - Package-based configuration; not cluster friendly
- Scalable transfer of computing environments
 - Grid'5000 [Cappllo05]
 - Pipelined data transfer for physical machine installation image

Agenda

- ✓ 1. Introduction
- ✓ 2. Optimization Algorithm
- ✓ 3. Architecture and Implementation
- ✓ 4. Experimental Evaluations
- ✓ 5. Related Work
- 6. Conclusions

Summary

- Proposed a novel virtual cluster installation technique that achieves:
 - **rapid installation** by the online adaptable virtual image caching
 - **scalability** by the near- $O(1)$ data transfer technique
 - **flexibility** by extending a (physical) cluster management system
- Achieved installation of 190 VMs in as fast as 40 seconds
 - Would be possible to install a 1000-VM virtual cluster only in two minutes!

Future Work

- More elaborate study of performance bottlenecks
 - Recent results with MPICH's bcast shows better scalability
- Model-based autonomous node selection for efficient and robust virtual cluster deployment
 - Automatically prunes less reliable, “slow” nodes
 - Optimal matchmaking between jobs and resources
 - CPU intensive: allocate nodes with faster CPUs
 - Network intensive: deploy onto nodes with faster interconnects
- Performance study on Grids
 - On the InTrigger nationwide distributed platform