USC INFORMATION SCIENCES INSTITUTE

Partitioning Large Workflows onto Multiple Sites with Storage Constraints

Weiwei Chen, Ewa Deelman

{wchen, deelman}@isi.edu

Information Sciences Institute, University of Southern California

pegasus

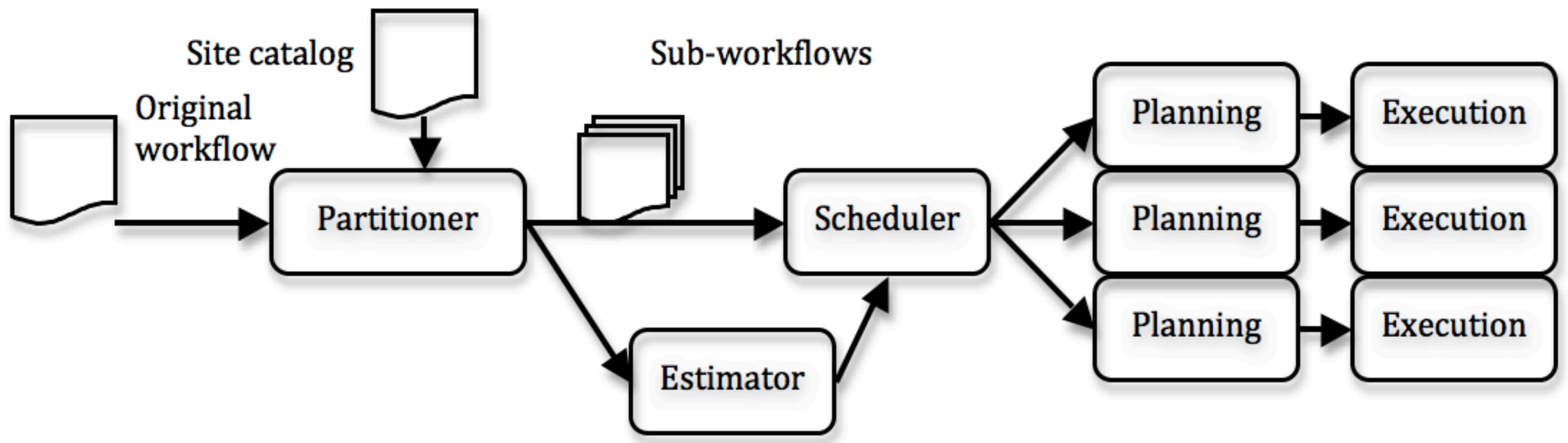
Problem Statement

•A Scientific Workflow describes the application components and their dependencies. Large-scale workflows require significant amount of storage and needs to use multiple execution sites and consider the storage constraints.

•We have developed a three-phase scheduling approach integrated with the Pegasus Workflow Management System to partition, estimate, and schedule workflows.

•Partitioning workflows into sub-workflows first reduces the complexity of the workflow mappings. The entire CyberShake workflow has 16,000 sub-workflows and each sub-workflow has more than 24,000 individual jobs.

Implementation



•Partitioner separates workflows into several sub-workflows within the storage constraints based on different heuristics.

•Estimator provides three methods to estimate the makespan of sub-workflows.

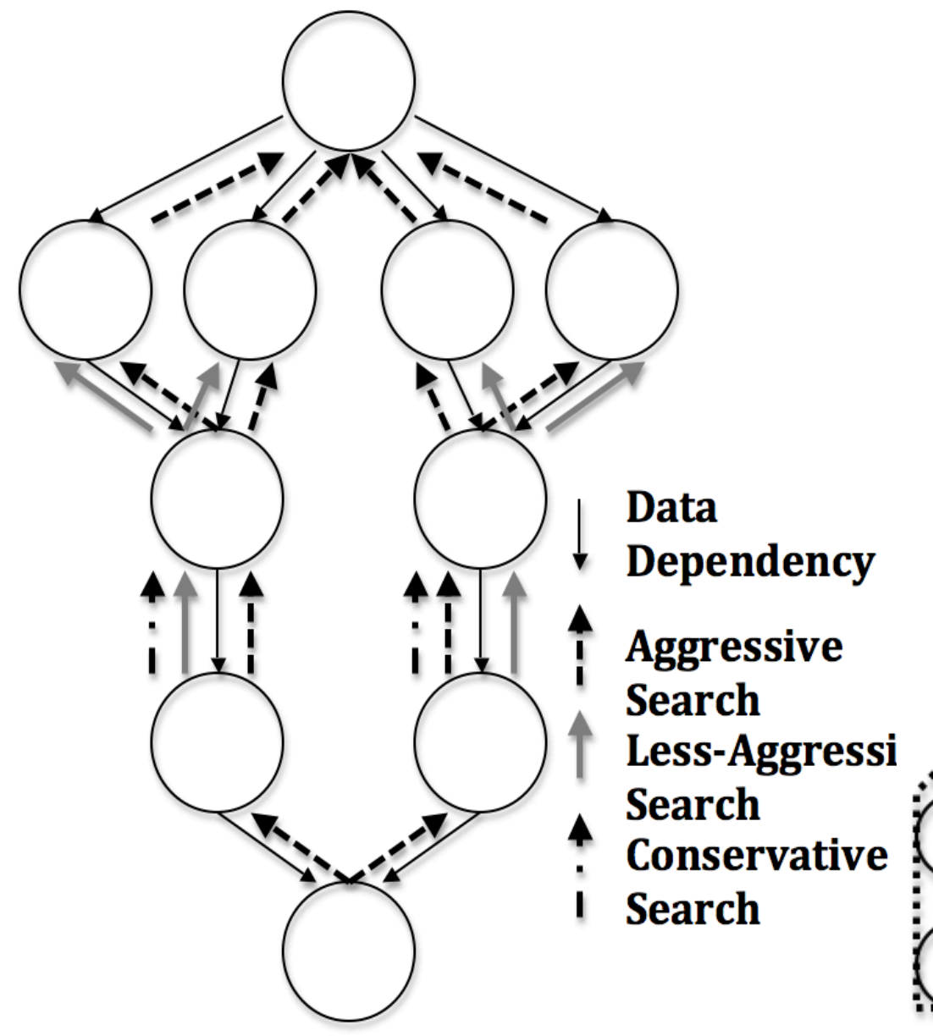
▪**Critical Path** is defined as the longest depth of the sub-workflow weighted by the runtime of each job.

▪**Average CPU Time** is the quotient of cumulative CPU time of all jobs divided by the number of available resources.

▪The **HEFT** method uses the calculated earliest finish time of the last sink job as makespan of sub-workflows assuming we use HEFT algorithm to schedule them.

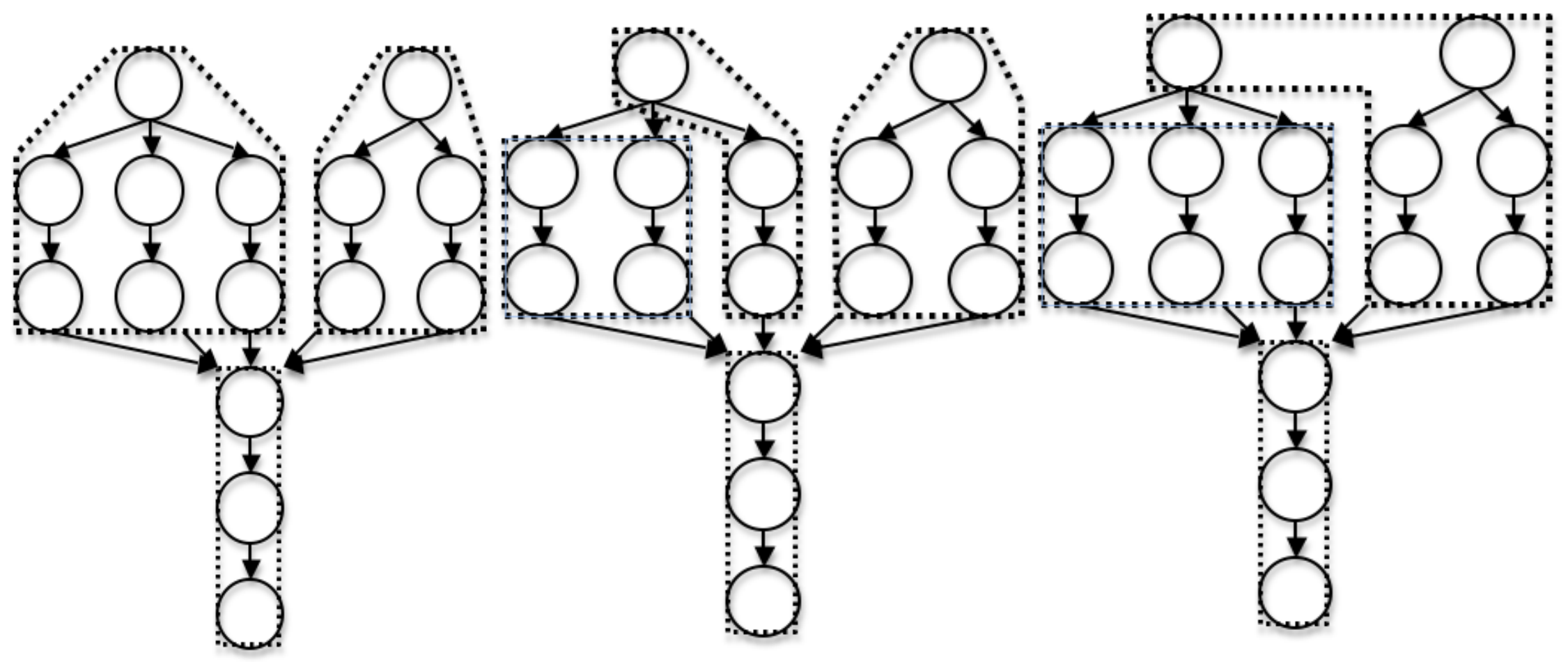
•Scheduler selects appropriate resources for the sub-workflows satisfying the storage constraints and optimizes the runtime performance. HEFT and MinMin scheduling algorithms are examined and compared.

Heuristics for Partitioning



Heuristic I: three steps of search

•Heuristic I tries three steps of search and adds the entire fan structure into a sub-workflow if it doesn't violate the storage constraint.



From Left to Right: Heuristic I, II, III

•Heuristic II adds a job to a sub-workflow if all of its unscheduled children can be added to that sub-workflow without causing cross dependencies or exceed the storage constraint.

•Heuristic III adds a job to a sub-workflow if each child of it has been scheduled and adding this job to the sub-workflow doesn't exceed the storage constraint.

Experiments

•Three workflows are examined on a cluster with 32 Condor slots and Glidein WMS is installed. We use Pegasus to plan the workflows and then submit them to Condor DAGMan that provides the execution engine.

▪Montage is an astronomy application that is used to construct large image mosaics of the sky. We ran the 8 degree square Montage case. It's I/O intensive.

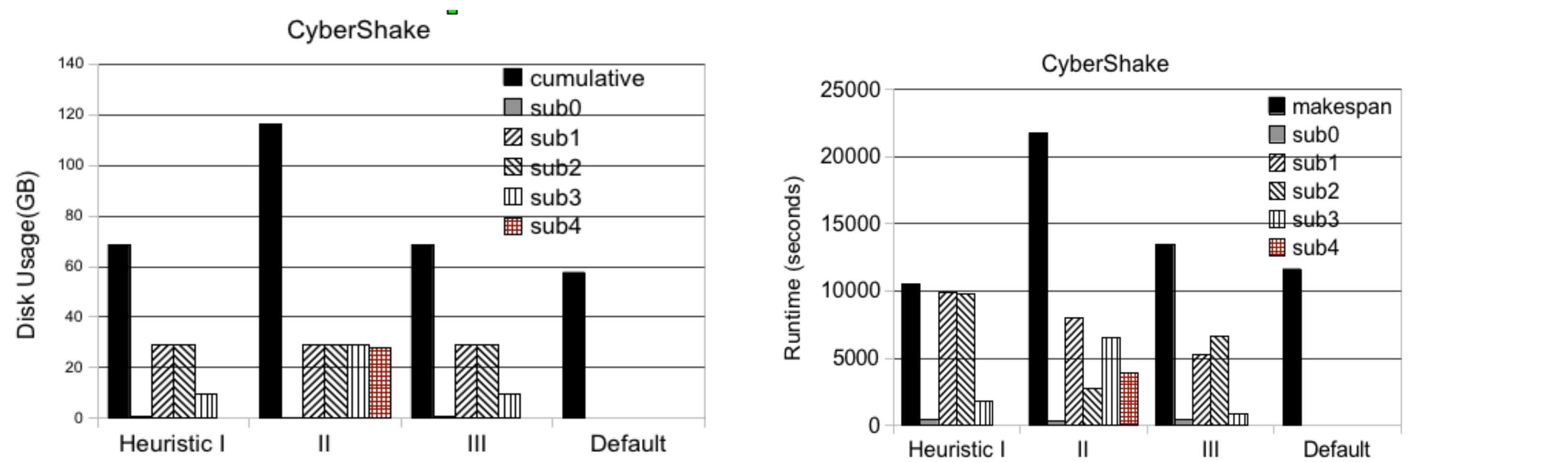
▪CyberShake calculates Probabilistic Seismic Hazard curves for several geographic sites in the Southern California area. We ran one partition of a geographic site. It's memory intensive.

▪Epigenomics maps short DNA segments collected with high-through gene sequencing machines to a reference genome. It's CPU intensive.

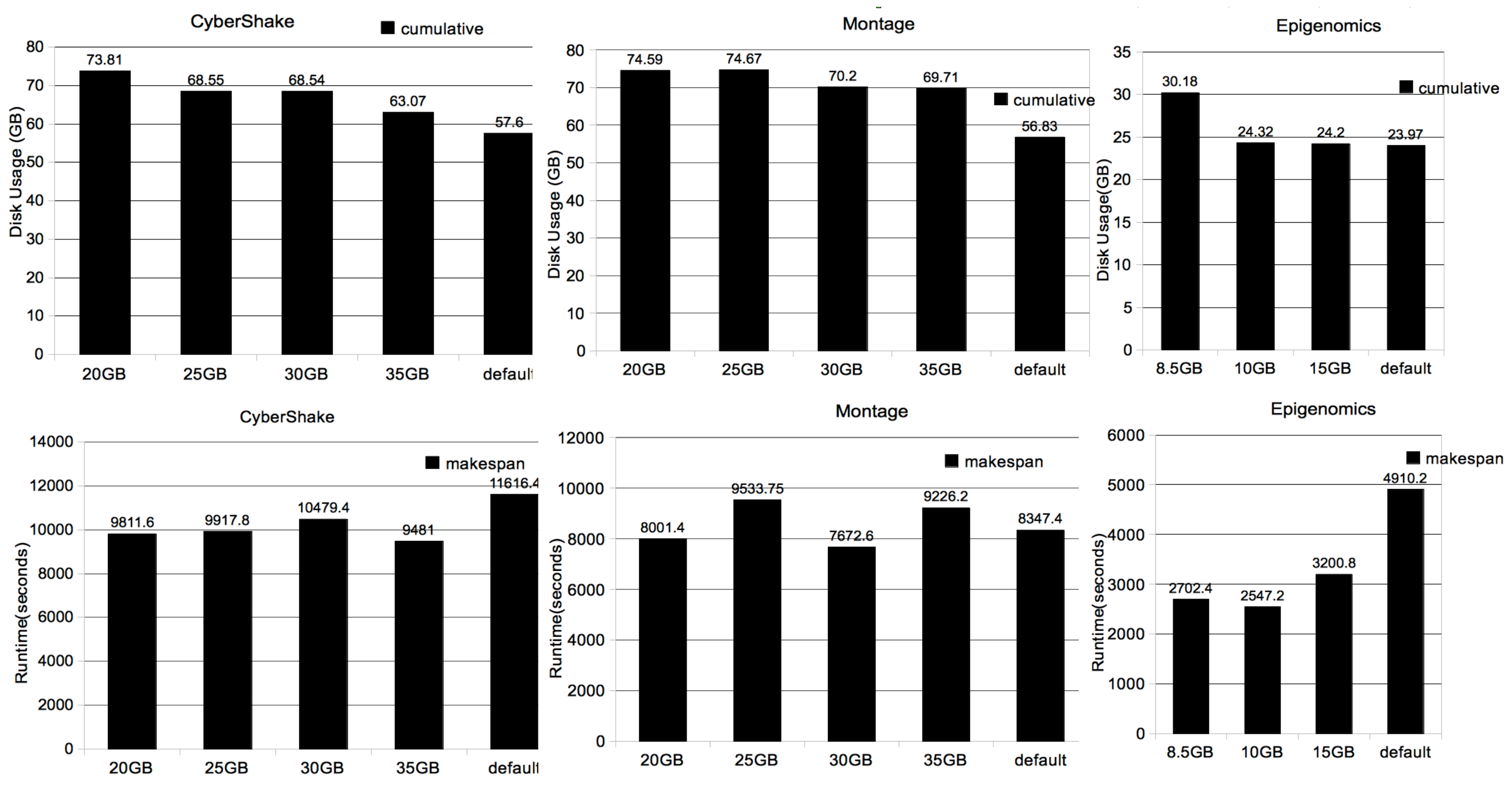
Performance

•Heuristic I, II, III improve the runtime by 9.79%, -15.87% and -86.86% compared to the default case running workflows on a single site.

•The reason is that Heuristic I avoids extra inter communication between sub-workflows



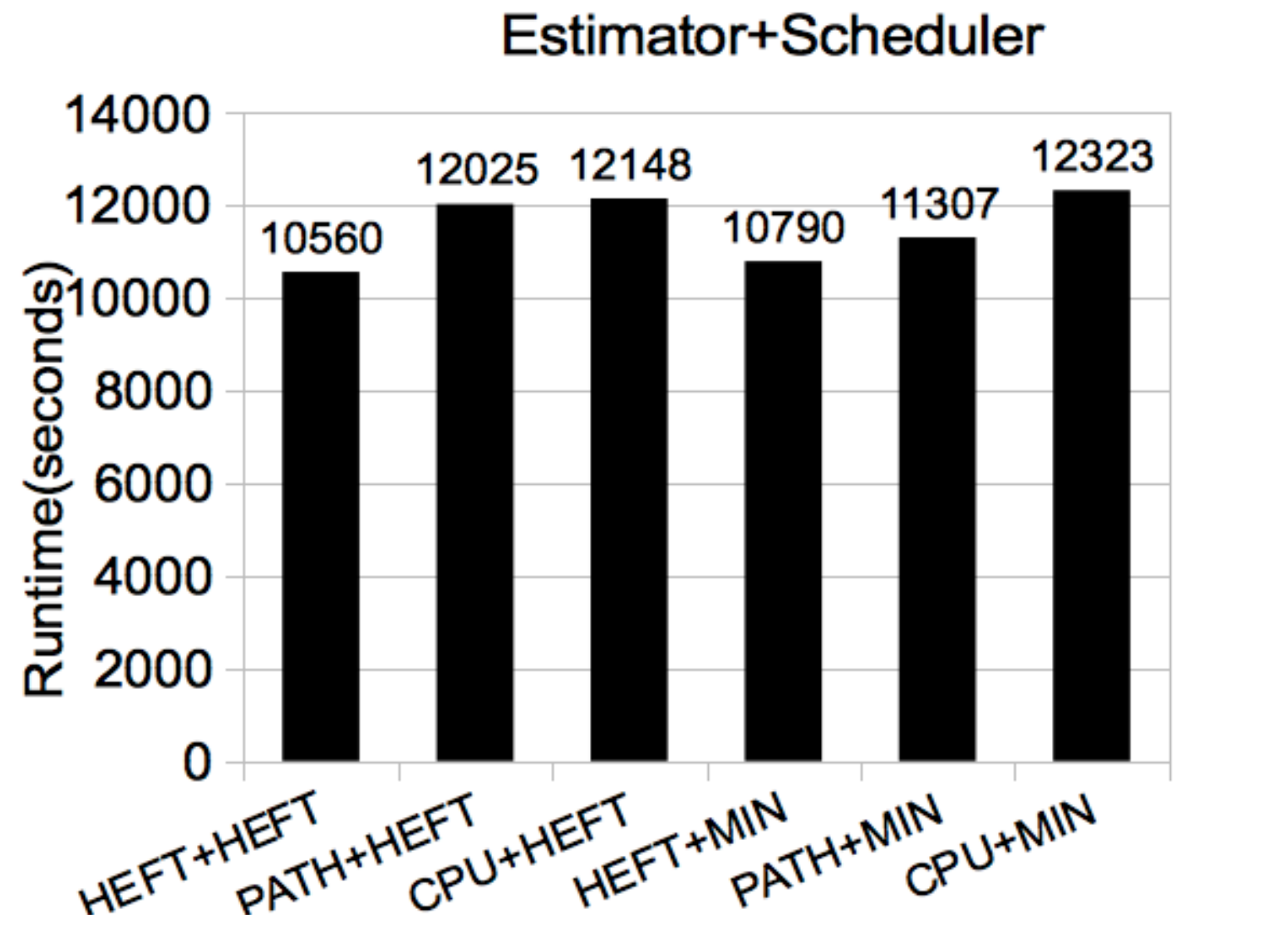
•Workflows across two or three sites perform the best in our experiments. More sites require more data transfers even though the computation part is improved.



•HEFT+HEFT improves the runtime by up to 14.3%.

•Average CPU Time and Critical Path don't consider the resource availability.

•HEFT scheduler is slightly better than MinMin since the number of sub-workflows has been reduced a lot.



This work is supported by NFS under grant # IIS-0905032. A research paper based on this work has been accepted by PPAM11. Details can be seen in www.isi.edu/~wchen/