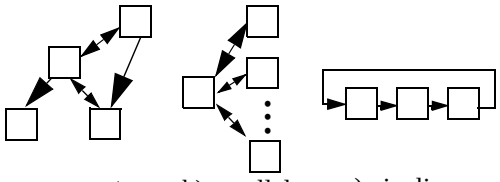
Metascheduling: A Scheduling Model for Metacomputing Systems

Jon B. Weissman
Division of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249
(jon@cs.utsa.edu)

Extended Abstract

Metacomputing is the seamless application of geographically-separated distributed computing resources to user applications. We consider the scheduling of *meta-applications*; applications consisting of multiple *components* that may communicate and interact over the course of the application. Components may be schedulable computations, remote servers or databases, remote instruments, humans-in-the-loop, etc. We divide meta-applications into three categories — *concurrent*, *parallel*, and *pipeline* (below). Concurrent is the classic meta-application in which a set of components each running in a single site are executing concurrently and exchanging data. Parallel is a special case of concurrent in which a component is replicated and distributed across multiple sites. Pipeline applications consist of components connected in a chain-like fashion.



a) concurrent b) parallel c) pipeline

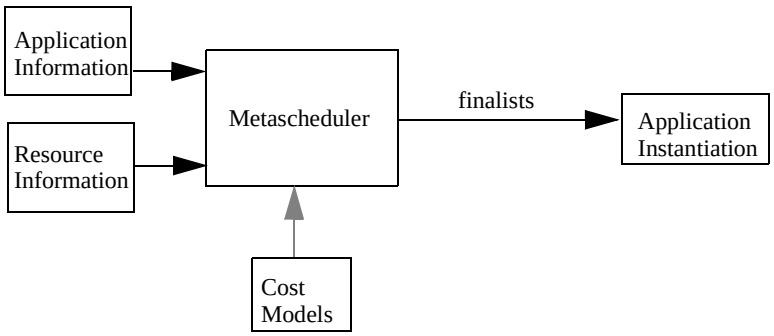
The scheduling of meta-applications across multiple sites, called *metascheduling*, is a complex problem especially when the network capacity is assumed to vary between sites. The metascheduler is a decision maker that computes candidate schedules that produce reduced completion time using application and site resource information (bottom). Cost models are used by the metascheduler to evaluate the potential set of candidate schedules for the application. The meta-application completion time T_{CT} is defined in terms of the computation and communication time for each component (given n components, m sites, where $1 \le (i \wedge j) \le n$ and $1 \le (k \wedge l) \le m$, C_i is the i^{th} component, S_k is the k^{th} site, and shown for pipeline and two types of concurrent meta-applications):

$$T_{CT}\,[\text{pipeline}] = \sum comp\_time\,[C_i, S_k] + comm\_time\,[C_i, C_j, S_k, S_l]$$

$$T_{CT}\,[\text{concurrent}] = \max\{comp\_time\,[C_i, S_k] + comm\_time\,[C_i, C_j, S_k, S_l]\}$$

$$T_{CT}\,[\text{concurrent, overlap within components}] = \max\{comp\_time\,[C_i, S_k], comm\_time\,[C_i, C_j, S_k, S_l]\}$$

The scheduling problem is to determine an assignment of components to site resources that minimizes T_{CT}. This scheduling problem is NP-complete and an exhaus-



name	value range/units	comments
num sites	3 .. 10	this size covers today's WAN testbeds
comp_rate	[1, 10000] MIPS	covers weak to powerful sites
intra_link_rate	[500, 1000] KBps -- fixed value	~ ethernet speed
inter_link_rate	[50, 100], [100, 200], [500, 10000] KBps	slow Internet up to vBNS-like speed
link_variance	1, 2, 5, 10	reflects ↑ network heterogeneity
affinity	1, 2, 5, 10	reflects ↑ component/site affinity
appl_type	concurrent w/wo overlap, pipeline	
num_components	3 .. 8	8 should cover most meta-apps
comp_amt	[10000, 100000], [100000, 100000], [1000000, 10000000] MInstructions	comp_ and comm_amt ranges cover medium-coarse grain apps
comm_amt	[1, 10000], [1, 100000], [1, 100000] Bytes	

tive search given *n* components and *m* sites has $O(m^n)$ complexity. We have developed a scalable scheduling heuristic with complexity $O(mn^2)$ applicable to meta-applications. The heuristic considers components in decreasing order of computational weight:

1. order application components by decreasing *comp_time* value
2. init PLACEMENT[C_i]= -1 for all components C_i
3. for each component C_i
4. for each site S_k
5. compute T_{CT} with PLACEMENT[C_i]=S_k, given PLACEMENT[C_j], j<i, unchanged
6. remember best T_{CT} and best associated site S_{best}
7. end for
8. PLACEMENT[C_i]=S_{best}
9. end for

We have run the scheduling heuristic over a large set of simulated metacomputing environments and meta-applications (above), and measured the performance of the heuristic with respect to the optimal schedule. The simulated metacomputing environment consists of a number of interconnected sites that provide a particular computation and communicate rate. Communication rates within the site and between sites are simulated. A link variance parameter is used to vary to inter-site communication performance allowing us to simulate a truly uneven network. Meta-applications consist of a type and a number of components. For each component, we have an amount of computation, and an amount of communication to all other components. These are varied to allow us to simulate a wide range of application granularities. We also provide an affinity parameter which is used to bias particular components to particular sites.

A simulation study of over 800,000 distinct environment and application instances was performed. The results that suggest that the heuristic performs very effectively over the parameter ranges (below). When the number of components is ≤ 5, the heuristic is within 10% of optimal on average. In general, it is within 20% on average. The heuristic also appears to be insensitive to the heterogeneity of the network environment.