Performance Model Based Scheduling of EMAN Workflows

Anirban Mandal[1], Ken Kennedy[1], Gabriel Marin[1], John Mellor-Crummey[1]

Charles Koelbel[1], Bo Liu[2], Lennart Johnsson[2]

[1]Dept of Computer Science, Rice University, Houston TX

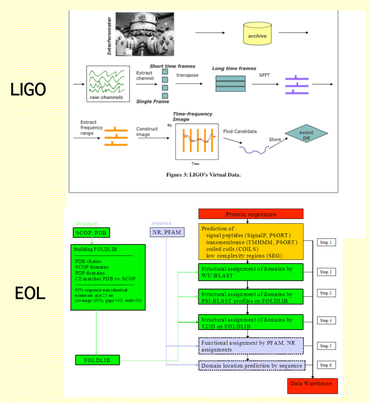
[2]Dept of Computer Science, University of Houston, Houston TX

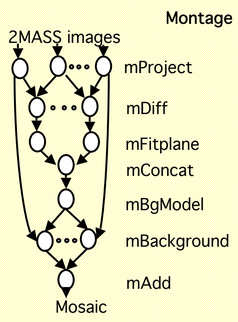
{anirban, ken, mgabi, johnmc, chk}@cs.rice.edu, {bliu2, johnsson}@cs.uh.edu

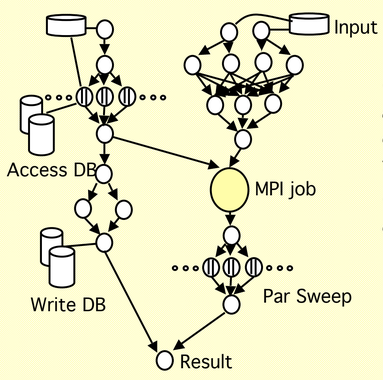
Abstract

We present our strategies of performance-model based, plan-ahead scheduling of workflows for a bio-imaging application called EMAN [Electron Micrograph Analysis]. We present our tools and strategies for constructing computational and memory-hierarchy performance models for EMAN workflow components. We then describe our algorithm for scheduling the workflow components onto Grid resources using the performance models. Results of our experiments show that our workflow scheduling strategies produce 1.5 to 2 times better makespan than existing strategies for this application. Also, we obtain good load balance across different Grid sites using these strategies.

Workflow Applications







- Application consists of a collection of components to be executed in a certain partial order for successful execution
- Several components with various data and control dependencies

Performance Modeling

- Rank of a component is total time to run it on a resource

$$Rank(comp_i, res_j) = EstExecTime_i(size(comp_i), arch(res_j)) + EstCommTime(comp_i, res_j)$$

- Communication time is latency plus bandwidth cost
  - Estimated from NWS

$$EstCommTime(c,r) = \sum_{p \in Parent(c)} \left(Lat(map(p), r) + Vol(p,c) \cdot BW(map(p), r) \right)$$

- Capture both the computation load and memory hierarchy performance
  - Memory hierarchy performance is a key for the application performance models
  - Including computation load modeling is necessary for the computation intensive application

$$EstExecTime(n,a) = \frac{FP(n,a) + L_1(n,a) + L_2(n,a) + L_3(n,a)}{Clock(a)}$$

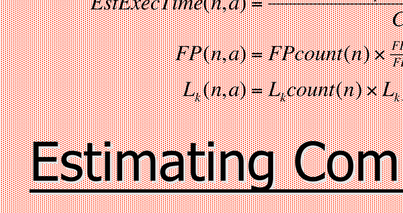
$$FP(n,a) = FPcount(n) \times \frac{FPdelay(a)}{FPpipes(a)}$$

$$L_k(n,a) = L_k count(n) \times L_k penalty(a), \quad k = 1,2,3$$

Estimating Computation Time

- (Floating point) Computation time is estimated from semi-empirical models
  - Form of model given by application experts
    - For eg. EMAN is floating-point intensive ⇒ Count floating-point ops
    - Key kernels are $O(n^2)$ ⇒ Fit to $c_2 \cdot n^2 + c_1 \cdot n + c_0$
  - Training runs with small data sizes
  - Collect floating-point operation counts from hardware performance counters
  - Least-squares fit of collected data to model to determine coefficients (FPcount)
  - Architecture parameters used to complete model (FPRpt)

Estimating Memory Access Time



- Memory access time (cache miss penalty) is estimated from black-box analysis of object code
  - Static analysis determines code structure
  - Training runs with instrumented binary produce architecture-independent memory reuse distance histograms
  - Fit polynomial models of reuse distances and number of accesses
  - Convolve with architecture features (e.g. cache size) for full model

Scheduling Workflows

- Problem: Map the components of the given Workflow DAG to a set of available Grid resources
- Objective function: minimize the makespan of the whole Workflow
- Overview
  - Resource Modeling using NWS and MDS
  - Sophisticated Application Component Performance models that take into account both computational performance and memory hierarchy performance
  - Walk the DAG and find the components that are currently available
  - Add data movement costs from the slowest predecessor in the performance model of the successor
  - Adapted known heuristics to schedule available components
- Step 1: Assign rank values
  - For each component, assign rank values for each resource
  - Rank values reflect the expected performance of a particular component on a particular resource
  - Convention: Lower the rank value, better is the match of the resource for the particular component
- Step 2: Solve for final mapping
  - Prepare a "Rank Matrix" from the rank values
  - Use heuristics to find out final assignments of components to resources

Scheduling Heuristics

- Used known heuristics in literature [parameter sweep applications] and adapt them. For a set of available components, choose the next (component-resource) mapping based on these heuristics
  - Min-min
    - Intuition: At each step minimally increase current makespan by choosing the next job having the minimum estimated completion time [over all jobs]
  - Max-min
    - Intuition: Satisfy long jobs first with the hope that the shorter ones are mapped in the same interval on other resources
  - Sufferage
    - Intuition: Give precedence to jobs who will suffer most if they are not assigned to their respective best resources

Algorithm

```
foreach heuristic do
    while all components not mapped do
        Find availComponents; // satisfy dependencies
        Calculate the rank matrix;
        findBestSchedule(availComponents, heuristic);
    endwhile
endforeach
Select mapping with minimum makespan among three;
Output selected mapping;
```

Overall Scheduling Algorithm

Algorithm 1. Workflow Scheduling

```
while all availComponents not mapped do
    foreach Component, j do
        foreach Resource, R do
            ECT(j,R)=rank(j,R)+EAT(R);
        endforeach
        Find minECT(j,R) over all R;
        Find 2nd-minECT(j,R) over all R;
    endforeach
    Calculate min(minECT(j,R)) over all j; //min-min
    Calculate max(minECT(j,R)) over all j; //max-min
    Calculate min(2nd-minECT(j,R)-minECT(j,R))
    over all j; //sufferage
    Store mapping;
    Update EAT(R) and makespan;
Endwhile
```

ECT(j,R): Estimated Completion Time of a component, j on a resource, R

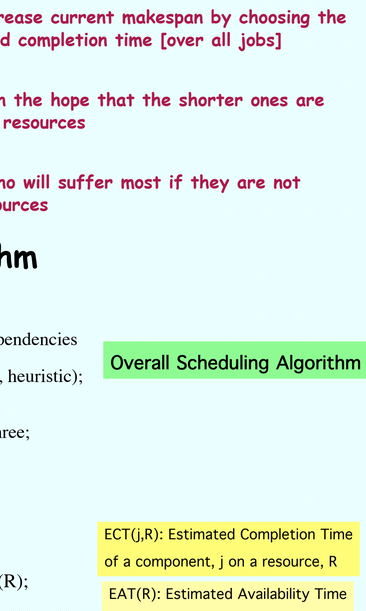
EAT(R): Estimated Availability Time of a resource, R

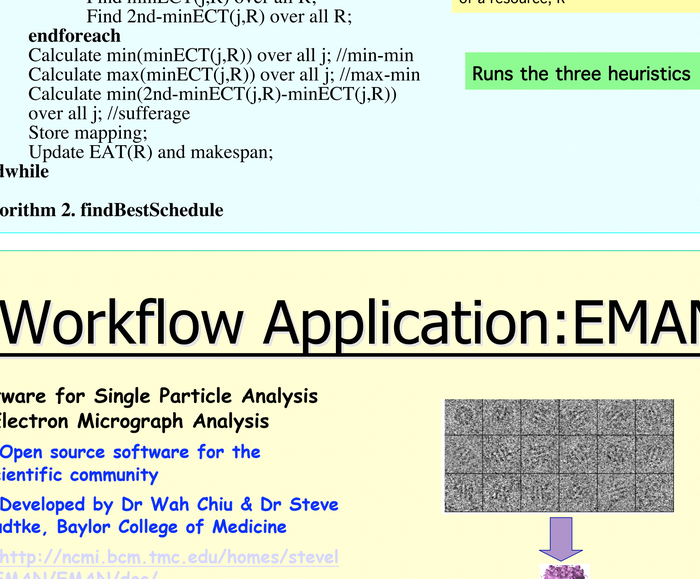
Runs the three heuristics

Algorithm 2. findBestSchedule

A Workflow Application:EMAN

- Software for Single Particle Analysis and Electron Micrograph Analysis
  - Open source software for the scientific community
  - Developed by Dr Wah Chiu & Dr Steve Ludtke, Baylor College of Medicine
  - http://ncmi.bcm.tmc.edu/homes/stevel/EMAN/EMAN/doc/
- Performs 3-D reconstruction of a particle from randomly-oriented images
  - Typical particle = Virus or ion channel
  - Typical images = Elctromicrographs
  - Typical data set = 10K-100K particles
  - Useful for particles about 10-1000nm
- GrADS/VGrADS project to put EMAN on Grid

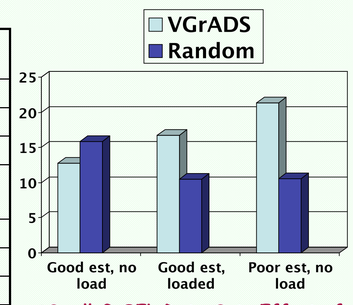




Results

Accurate Load Balance

Component Name	Resource(s) Chosen	# instances	Component Exec. Time
proc3d	i2-58	1	<1 min
project3d	i2-58	1	1h. 48 min
proc2d	i2-58	1	<1 min
classesbymra	i2-53 to 58 torc1-7	68 [i2-*] 42 [torc*]	84 h. 30 min 81 h. 41 min
classalign2	i2-53 to 58	379	45 min
make3d	i2-58	1	47 min
proc3d	i2-58	1	<1 min
proc3d	i2-58	1	<1 min

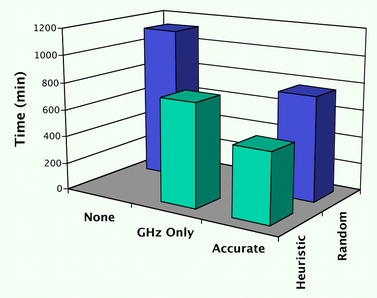


- Small GrOEL Data Set: Effect of Machine loads

Value of Performance Models and Heuristics

	i(RTC) IA-64	i(medusa) Opteron	n(RTC)	n(medusa)	t(RTC) (min)	t(medusa) (min)	makespan (min)
RNP	89	21	43	9	1121	298	1121
RAP	57	53	34	10	762	530	762
HGP	58	52	50	13	757	410	757
HAP	50	60	50	13	386	505	505

- Set of resources:
  - 50 rtc nodes at Rice (IA-64)
  - 13 medusa nodes at U of Houston (Opteron)
- RDV data set: medium/large (2GB)
- Varying scheduling strategy
  - RNP - Random / No PerfModel
  - RAP - Random / Accurate PerfModel
  - HCP - Heuristic / GHz Only PerfModel
  - HAP - Heuristic / Accurate PerfModel

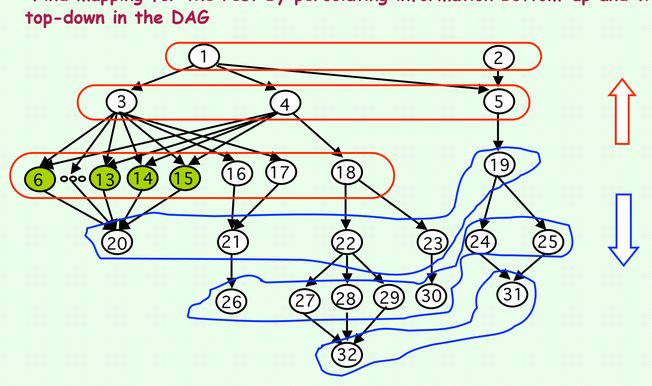


Accuracy of Performance Models

- Our performance models were accurate
  - Good Case
    - rank[RTC_node] / rank[medusa_node] = 3.41
    - actual_exec_time[RTC_node] / actual_exec_time[medusa_node] = 3.82
  - Less good Case
    - rank[acrl_node] / rank[medusa_node] = 2.36
    - actual_exec_time[acrl_node] /actual_exec_time[medusa_node] = 3.01
- Accurate relative performance model values result in efficient load balance of the classesbymra instances

Future Work

- Proposed improvements
  - If expected data movement cost between two components is very high, they should be mapped to the same resource - "fusion" brings the DAG to the "right" granularity
  - The approach in the data movement reduction step is to model it as a "weighted-fusion" problem
  - Generally, some application components are key components in the DAG and the makespan of the DAG heavily depends on whether the key steps have been mapped to the "right" set of resources or not
  - The approach in the global scheduling step is
    - Conditionally map key components
    - Find mapping for the rest by percolating information bottom-up and then top-down in the DAG



References

- "Scheduling Strategies for Mapping Application Workflows onto the Grid" Mandal, A, Kennedy, K, Koelbel, C, Marin, G, Liu, B, and Johnsson, L 14th IEEE Symposium on High Performance Distributed Computing (HPDC 2005). IEEE Computer Society Press.
- "Resource Allocation Strategies for Workflows in Grids" Blythe, J, Jain, S, Deelman, E, Gil, Y, Vahi, K, Mandal, A, and Kennedy, K IEEE International Symposium on Cluster Computing and the Grid (CCGrid05)
- "Heuristics for Scheduling Parameter Sweep applications in Grid environments" Henri Casanova, Arnaud Legrand, Dmitrii Zagorodnov and Francine Berman Proceedings of the 9th Heterogeneous Computing workshop (HCW'2000), 2000.

Acknowledgments

- This work is supported by the National Science Foundation under Grant No.ACI 0103759(the GrADS Project) and Cooperative Agreement No. CCR-0331654 (the VGrADS Project). This work was supported in part by the Rice Terascale Cluster funded by NSF under Grant EIA-0216467, Intel, and HP.
- We sincerely acknowledge the help we received from Dr. Wah Chiu and Dr. Steve Ludtke from the Baylor College of Medicine and Anshu Dasgupta, Mark Mazina and Dr. Keith Cooper from Rice University.