



ClustalW Optimization: Adaptive Scheduling

Presented by

Chung Shin Yee

chungsy@ihpc.a-star.edu.sg

Outline

- Motivation & Objectives
- Overview of ClustalW
- Preliminary Study
- Summary of Issues
- Adaptive Scheduling
- Appendix

Motivation

- ClustalW is commonly used software in bio-informatics.
- Compute intensive.
- Currently do not leverage cluster computing resource efficiently.
- Good case study in performance optimization.

Objectives

- Analyze efficiency of different workloads.
- Reduce overall turn around time
- Increase system throughput & utilization.

Overview of ClustalW

- 3 major phases.

1) Pairwise alignment.

- Compute intensive.
- High parallelism.

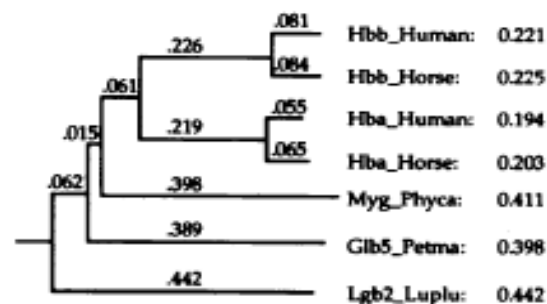
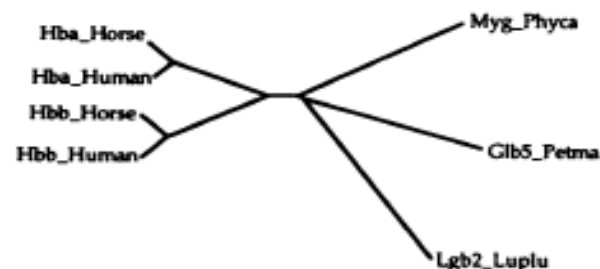
2) Guide-tree generation.

- Limited computation.

3) Progressive alignment.

- Compute intensive.
- Limited parallelism.

Hbb_Human	1	-				
Hbb_Horse	2	.17	-			
Hba_Human	3	.59	.60	-		
Hba_Horse	4	.59	.59	.13	-	
Myg_Phyca	5	.77	.77	.75	.75	-
Glb5_Petma	6	.81	.82	.73	.74	.80
Lgb2_Luplu	7	.87	.86	.86	.88	.93
		1	2	3	4	5
						6

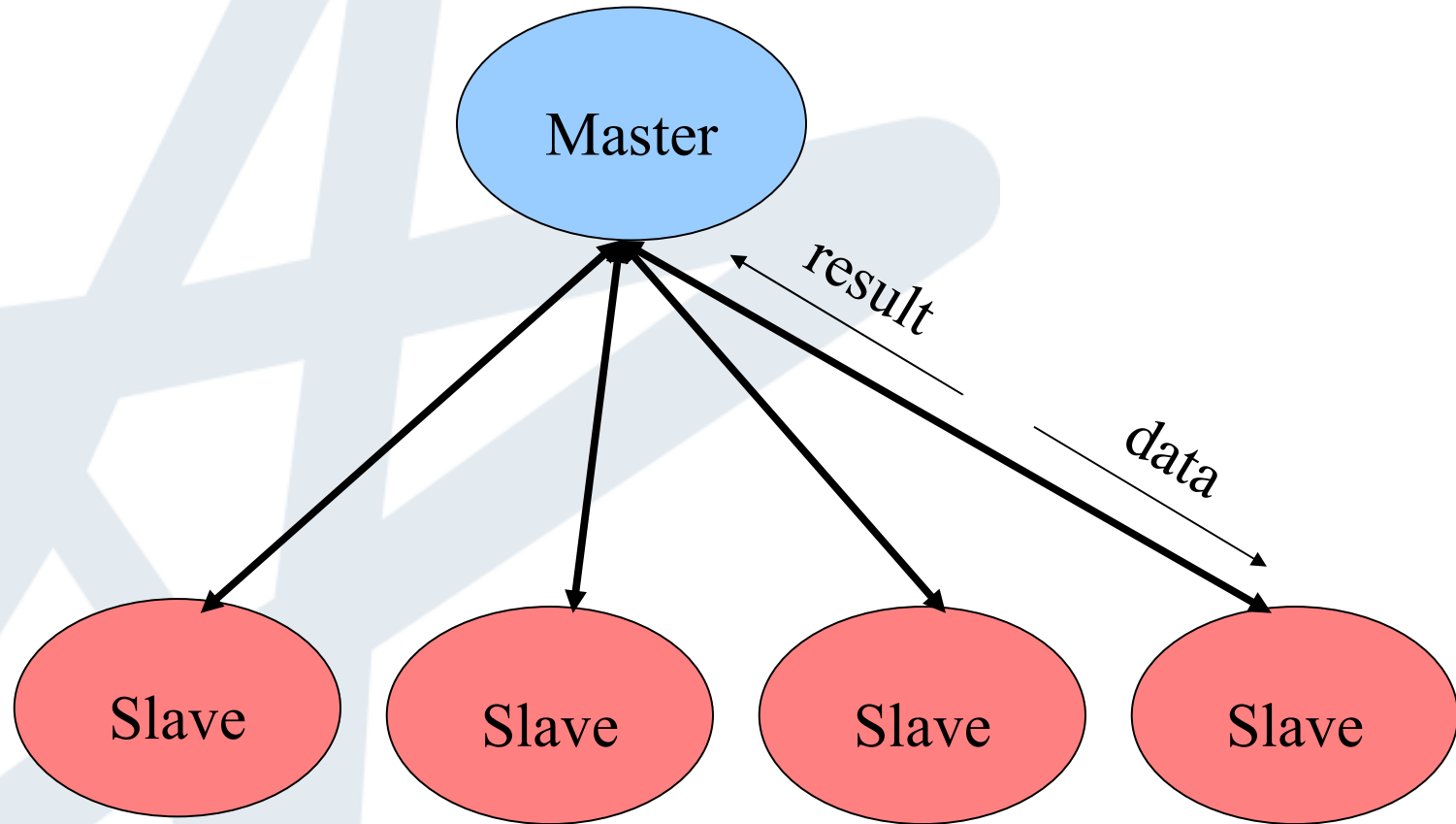


```

--VHLVPEEKAVPTALNGKVI--DDEVQOHALGRLLVVETPTQVFFSFGDLST
--VQLDDEEKAAVLALNDKVI--EEVVOGHALGRLLVVETPTQVFFSFGDLSEH
--VLSADKTYVKAANGKVDGAGAGTYGAAALEKMFLEPTTKYFFPEFDLS--
--VLSADKTYVKAANGKVDGAGAGTYGAAALEKMFLEPTTKYFFPEFDLS--
--VLSADKTYVKAANGKVDGAGAGTYGAAALEKMFLEPTTKYFFPEFDLS--
IVAPLGAARKYKIRANAPVYVSTVETSOVDILVETFTSFAAQHFFPKFKGLTT
--GALVDSQAAVLESEKEMKAMKPKTRVFFLLVLEIAPAAKILFSPFLKOTSE

```

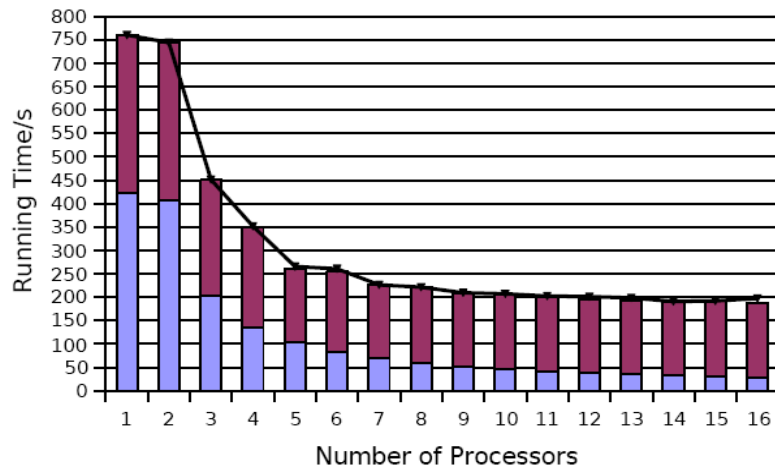
Master & Slaves Paradigm



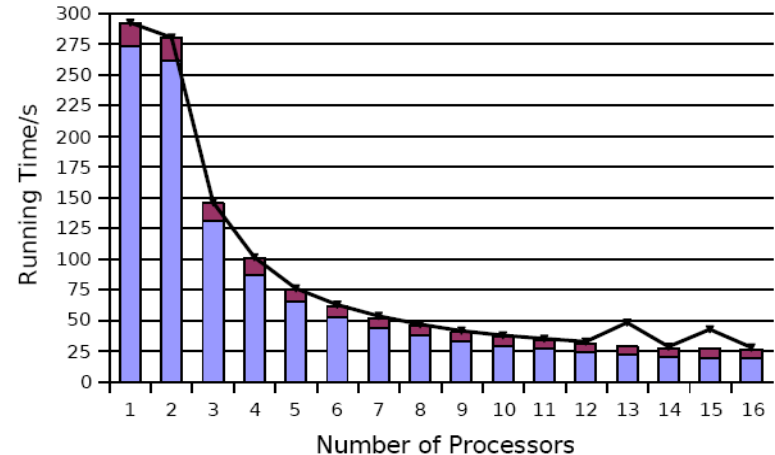
Preliminary Study

- Degree of parallelism changes in different phases.
 - Dynamic processor requirement.
 - More processors not necessary faster.

Running Time (Case: HIV)

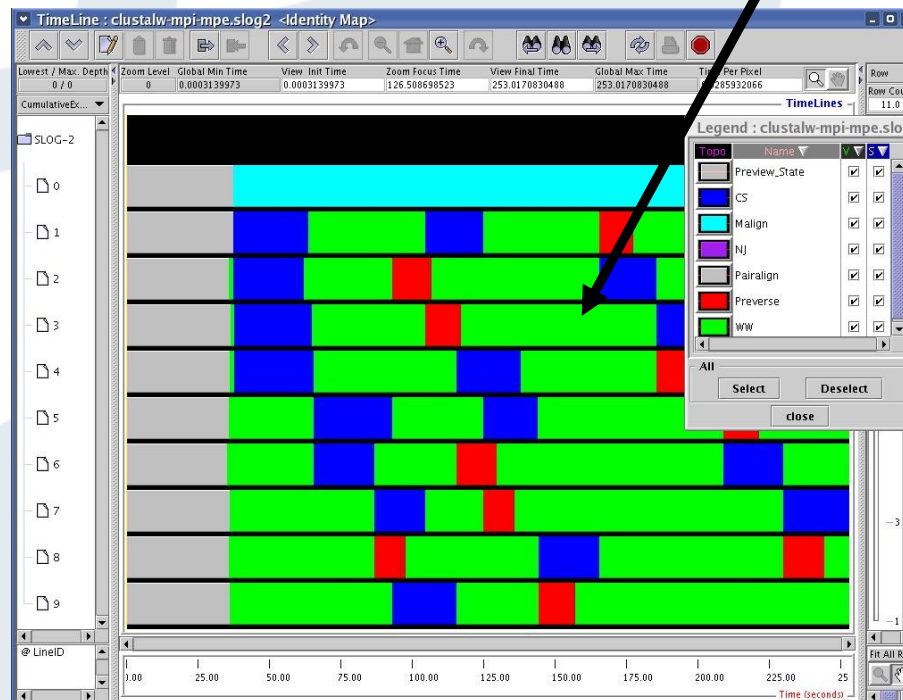


Running Time (Case: pfam)



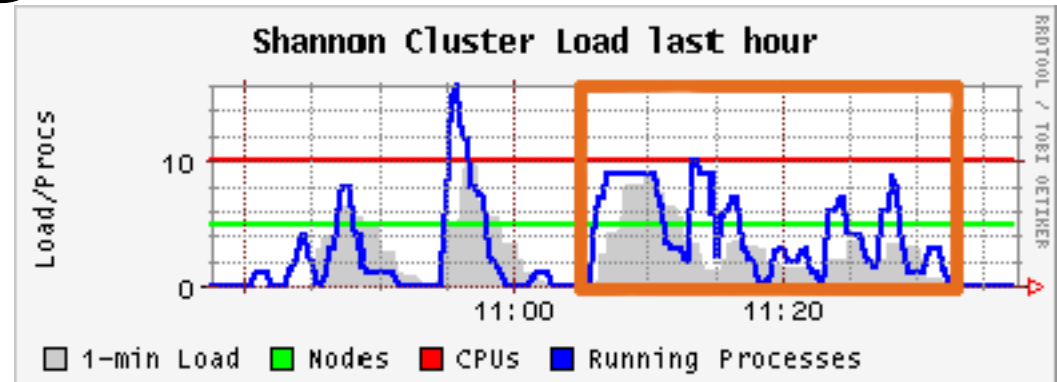
Profiling using Jumpshot

- Allocated processors could be **idling**.
- Some execution hot spots have been identified.

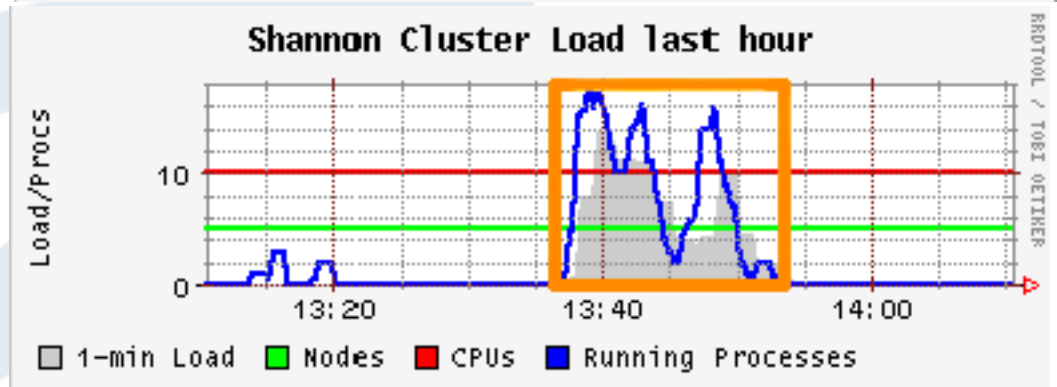


Running 8 Cases in Batch

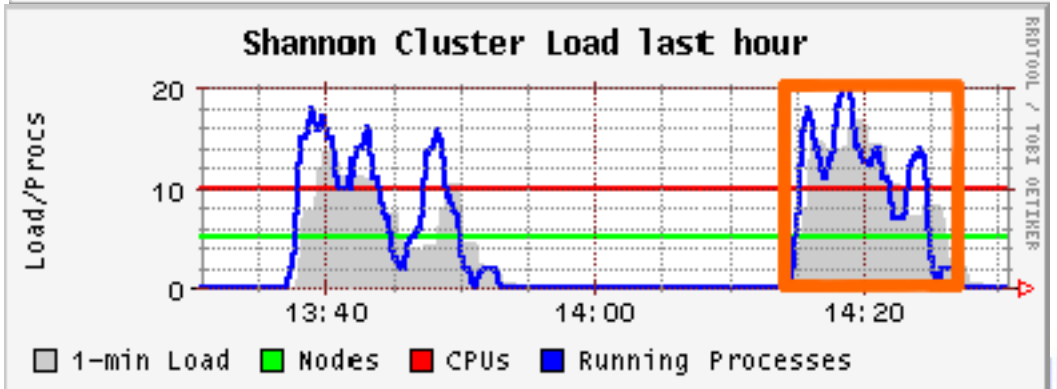
1 Job at a time



2 Jobs at a time

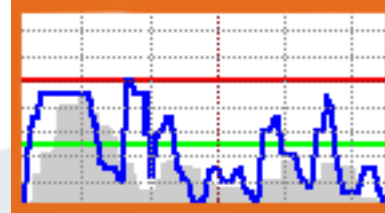


3 Jobs at a time

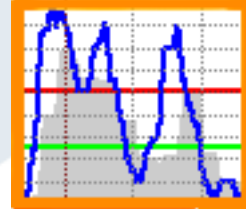


Comparing Cluster Load

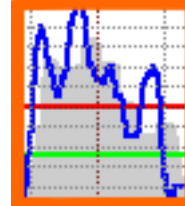
1 Job at a time



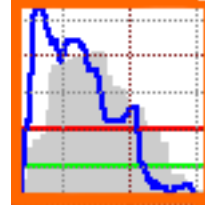
2 Jobs at a time



3 Jobs at a time



4 Jobs at a time



Summary of Issues

- Static processor allocation could waste processor cycles, or limit the performance.
- How many processors should I request?
- Internal scheduling may conflict with other jobs, sharing the processors.
- Context switching could increase running time by 25%.

Adaptive Scheduling

- Use local information to perform task scheduling and maintain scalability.
- Share a single pool of processors efficiently.
- Use minimum global information.
- Periodically estimate the number of processors required, k_e , based on effective utilization.

Adaptive Scheduling Strategy

- 1) Initially number of active processes, $A = 1$.
- 2) **If** ($k_e < A - \varepsilon$ AND $A \geq 2$) **Then**
Deactivate $\max(1, (A - k_e) / \delta)$ least utilized processors.
- 3) **Else If** ($k_e \approx A$) **Then**
Activate $\min(k - A, (A \times \sigma) - A)$ randomly chosen.
- 4) **Else** “do nothing”

Thank You

Comments?

Chung Shin Yee

chungsy@ihpc.a-star.edu.sg

*Having good hardware is great,
we need good software too!*

Cluster: Hardware & Software

- 5 nodes x 2 Itanium II 1.4GHz
- 2/4GB RAM
- 100-based Ethernet, Myrinet 2000 SCI
- Intel C++ Compiler 9.0
- MPICH 1.2.6/1.2.7

SMP: Hardware & Software

- Altix 128 x Itanium II 1.5GHz
- 512GB RAM
- NUMALink
- Intel C++ Compiler 9.0
- SGI MPT (MPI library)