Small World Asynchronous Parallel Model for Genome Assembly

Jintao Meng, Bingqiang Wang, Yanjie Wei, Jiefeng cheng, Shengzhong Feng, Pavan Balaji

Shenzhen Institutes of Advanced Technology, CAS

Email：{jt.meng, yj.wei, jf.cheng, sz.feng}@siat.ac.cn

Introduction

There is a growing gap between the output of new generation of massively parallel sequencing machines and the ability to process and analyze the resulting data. Sequencing throughput has recently been increasing at a rate of about 5-fold per year, while computer power follows "Moore' Law," doubling only every 18 or 24 months. Single server's computation power has been already exhausted, parallel applications on supercomputers are the only technology to handle big data in bioinformatics. Scalability, however, is the primary metric to prove its feasibility.[1]

State-of-the-art trials on scalable assembler include ABySS[2], YAGA[3], and Hecate[4]. The first two share the same computational model, BSP, and the last one adapts MapReduce platform. Experimental results shows that ABySS, Hecate and YAGA, can scaled to 64, 96, 256 cores, respectively.

Methods

A scalable assembly strategy and computational model are equally important for a highly scalable and efficient parallel assembler. The main contributions of this work are listed as below:

- **Multi-step bi-directed graph (MSG)**

1. Multi-step bi-directed graph of a given string set *S is:*

$$G_k(S) = \{V_S, E_S\} = \{\bigcup_{1\leqslant i\leqslant h} V_{s_i}, \bigcup_{1\leqslant j\leqslant g} (\bigcup_{1\leqslant i\leqslant h} E_{s_i}{}^{j})\}.$$

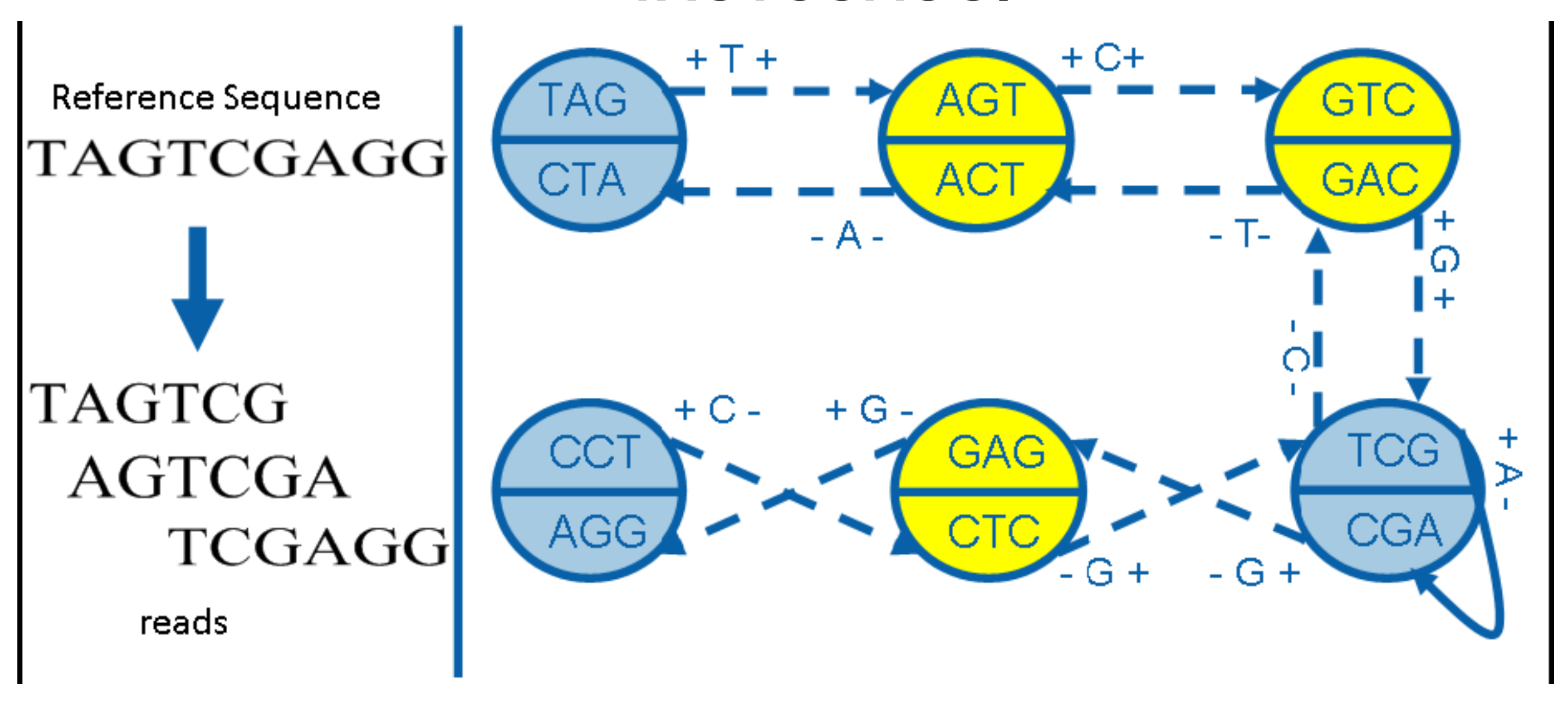
MSG is the first scalable mathematical abstraction of genome assembly. MSG has two propertys:

a). full-extended edges E_s* are contigs,

b). edge merging operation $\oplus$ is associative , Q(E_sV**0**, $\oplus$) is a semigoup.

here, *a* grantees the equivalence of contigs and full-extended edges, and *b* bridges the genome assembly problem with semi-group computing.

Fig 1. An example of MSG generated by three reads of Reference TAGTCGAGG.



2. **Small World Asynchronous Parallel Model (SWAP)**

SWAP is designed primarily for computation in semigroup. The schedule of SWAP is:

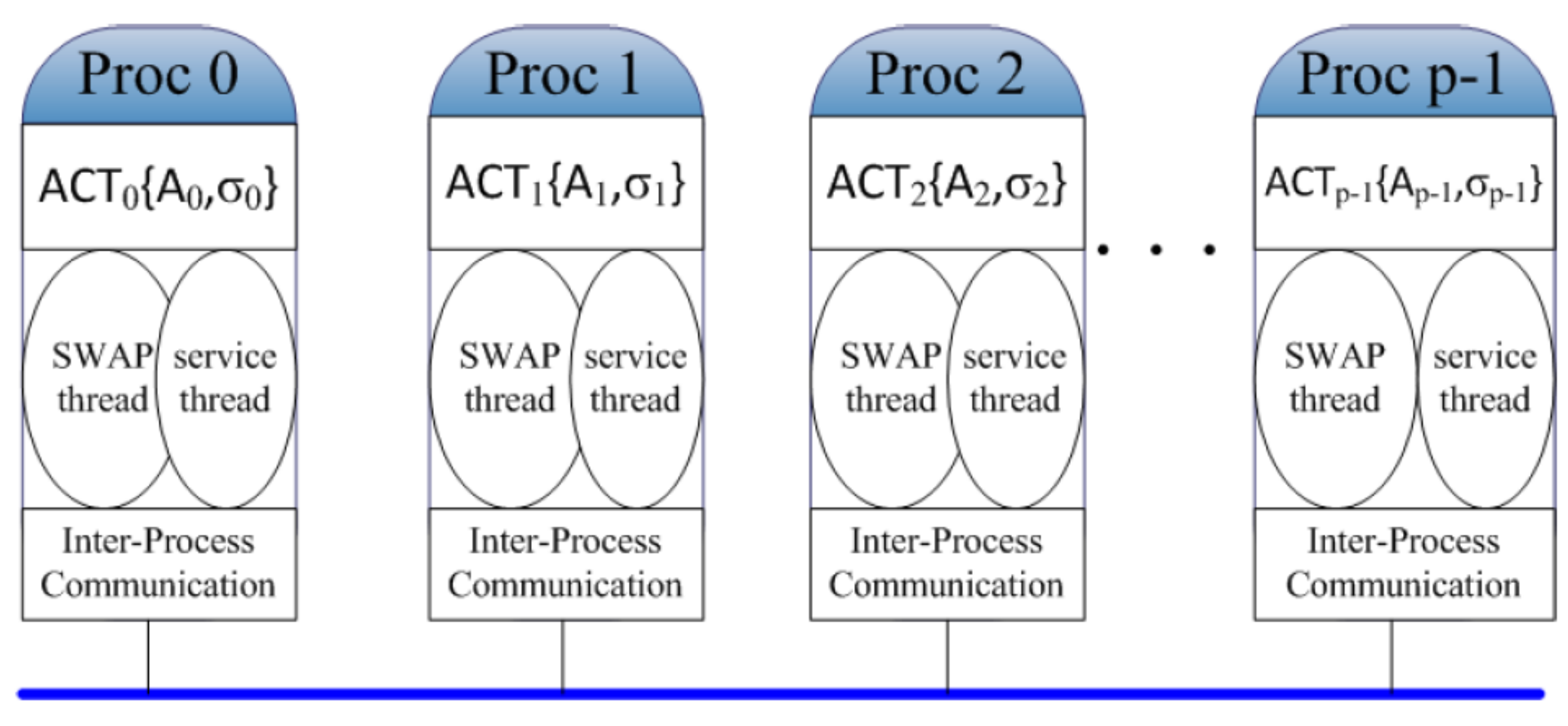
a). Lock action is applied to lock operation (*a*, *b*) and elements *a*, *b*.

b). Computing will be performed for operation (*a*, *b*), and the values of *a*, *b* are updated accordingly.

c). Unlock action will be triggered after computing step to release operation (*a*,*b*) and *a*, *b*.

SWAP makes a tradeoff between BSP and LogP with small world synchronization instead of global synchronization in BSP and peer synchronization in LogP.

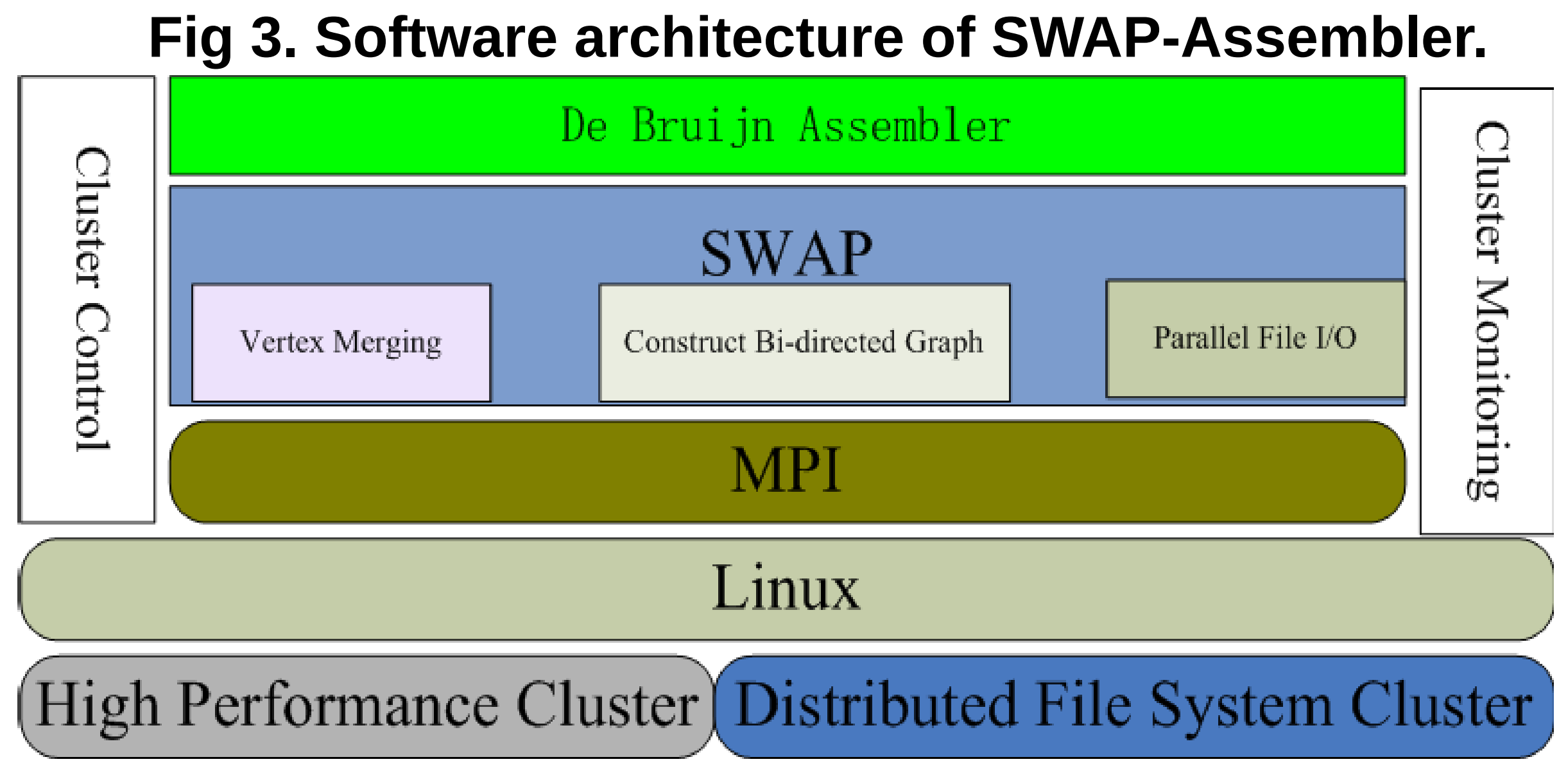
Fig 2. The implementation of SWAP on cluster.



3. **SWAP-Assembler**

SWAP-Assembler is an asynchronous parallel framework developed for large genome assembling over a community of computers using SWAP model. Given the number of processors p, the complexity of SWAP-Assembler is *O(n/p)* parallel computing time, *O(n log log (g)/p)* communication volume, and *O(log log(g))* communication round. Here g is the length of genome, and *n* is the number of nucleotides in all input reads.

Fig 3. Software architecture of SWAP-Assembler.



Results

We use a Perl script to generate the following two datasets: 50x coverage of Yeast chromosomes containing 17 million reads, and 50x coverage of C.elegans chromosomes containing 141 million reads. The error rate is set to be 1%, and the length of reads ranges from 36bp to 50bp. The primary goal of this experiment is to demonstrate the scalability of SWAP model on handling large-scale graphs using parallel system with distributed memory.

The runtime of SWAP-Assembler on Yeast dataset is displayed in Figure 4,and the time usage is divided into three phases, Parallel File I/O, graph construction, and edge merging. The runtime is dominated by the third phase, where hundreds of processors are locking their neighbors, merging edges, deleting semi-extended nodes and edges, and unlocking their neighbors. Figure 4 shows that this phase has good scalability. The speedup is about 50 when the number of processor scales from 10 to 640 and the overall runtime of assembler on Yeast dataset is reduced by a factor of 30.

For C.elegants dataset, Figure 5. shows the total speedup is 20 when the number of processors scales from 10 to 640.

Fig 4. Time usage on Yeast dataset.

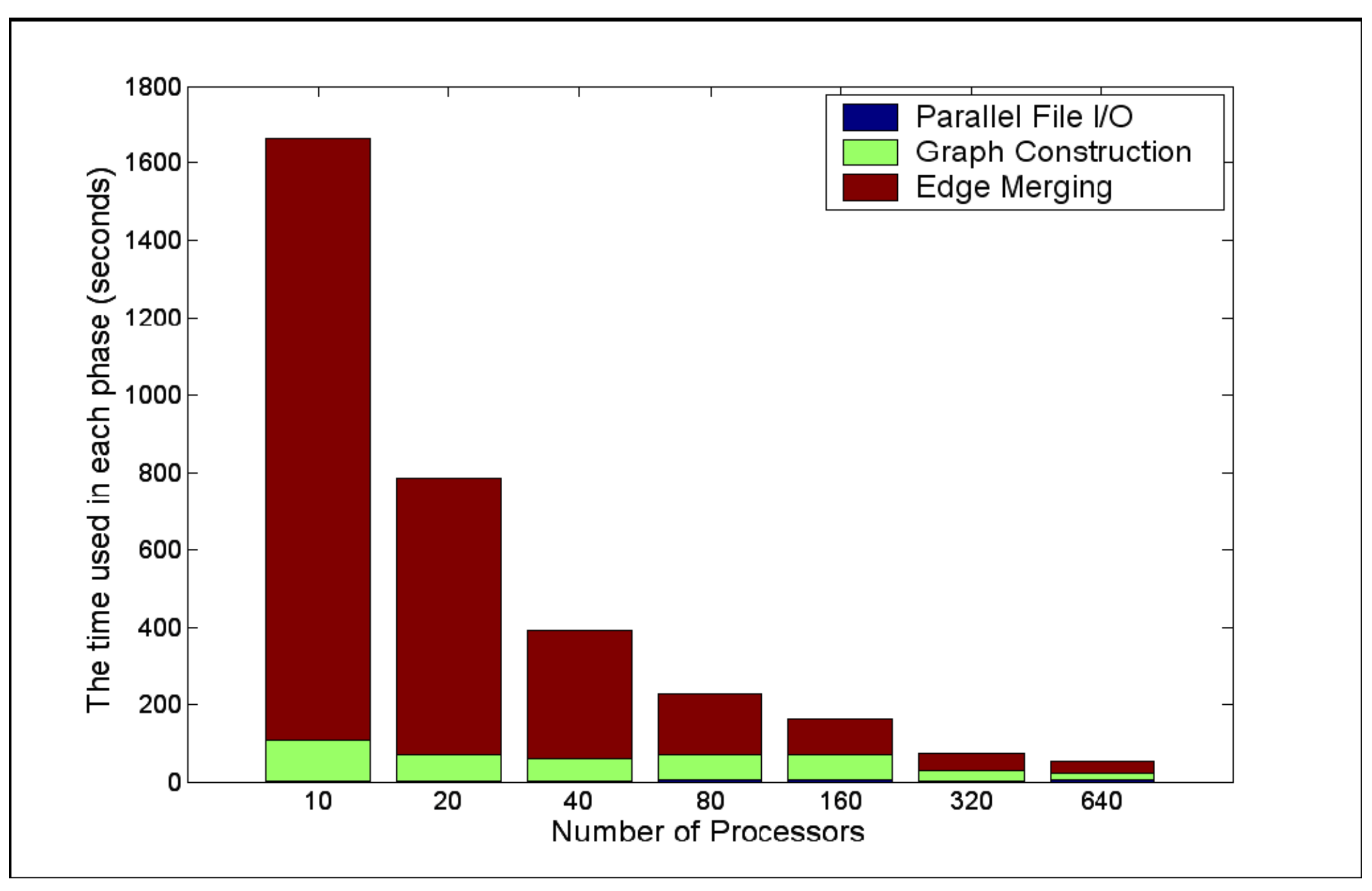
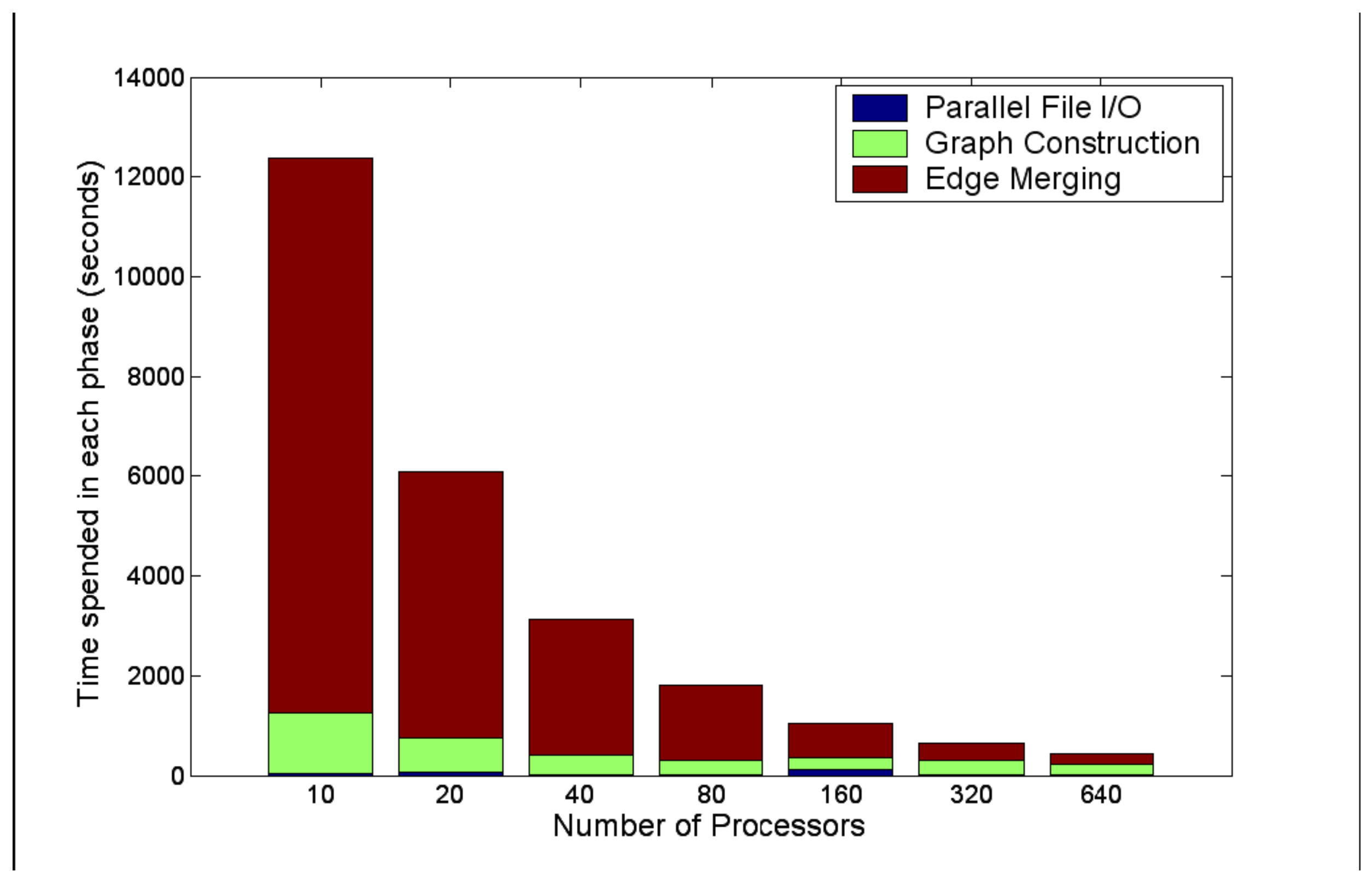


Fig 5. Time usage on C.elegans dataset.



Conclusions

In this paper, Our contributions are list as below:

- ◆ Abstract the problem of genome assembly using MSG, and the unanimous path compaction problem was transformed to merge semi-extended edges into full-extended edges.
- ◆ The proposed SWAP computation model introduced a local synchronization and global asynchronization mechanism to maximize the parallelism on computing semigroup.
- ◆ Based on SWAP model, we developed a new assembler framework named SWAP-Assembler.
- ◆ Simulation shows that when the number of processors scales from 10 to 640, a factor of 30 and 20 speedup is achieved on assembling Yeast and C.elegans datasets, respectively.

References:

1. Jintao Meng, Yanjie Wei, Jiefeng Cheng, Shengzhong Feng, “Small World Asynchronous Parallel Model for Genome Assembly, In NPC 2012, Kwangju, Korea.