JailDocker: Docker Container-based Scalable Partitioning for Apache Spark Scale-Up Server

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings. It is an *alternate* style which produces a *tighter-looking* paper and was designed in response to concerns expressed, by authors, over page-budgets. It complements the document *Author's (Alternate) Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file. A compiled PDF version is available on the web page to help you with the 'look and feel'.

CCS Concepts

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Keywords

ACM proceedings; LaTeX; text tagging

1. INTRODUCTION

HPC를 위한 스파크가 필요하다. 리눅스 운영체제가 많이 사용된다. 기존 연구들은 HPC에 spark를 적용하기 위한 방법들이 연구되고 있다. 15코어 이하에서 확장성을 검사하였음. 하지만, 이러한 연구들은 리눅스 운영체제의 특징을 활용하지 못하는 문제점이 있다. 리눅스 운영체제 확장성에 대해서 가장 중요한것은 리눅스는 conflit free한 운영체제가 아니다. 이유는 !!! 여러가지 상황을 고려하였기 때문에 만약 응용프로그램이 scalalabe하게 디자인이 되어 있다면, 리눅스라도 스케일러블한 상황으로 만들 수 있다[].

XXXXX 'XXXX', USA

DOI: XX.XXX/XXX_X

스케일러블 한 Scale-Up 서버를 위한 연구들이 진행되고 있지만,

Scalability 문제를 해결하기 위해, 본 연구는 메모리 파티션 기반의 JailDocker를 만들었다. JailDocker는 Scale-Up server에서 최적의 성능을 내기 위한 파티션 방법이다. 기반을 사용하기 우리는 Docker를 사용하였다.

2. SPARK SCALABILITY

2.1 Benchmarks

2.2 Scalability

Figure 2(a)) since it waits to acquire

2.3 CPU Utilization

2.4 Memory usage

2.5 Lock

3. PARTITIONING

Kubernetes

4. EVALUATION

This section answers the following questions experimentally:

- Does LDU's design matter for applications?
- Why does LDU's scheme scale well?
- What about LDU's read-write ratio?

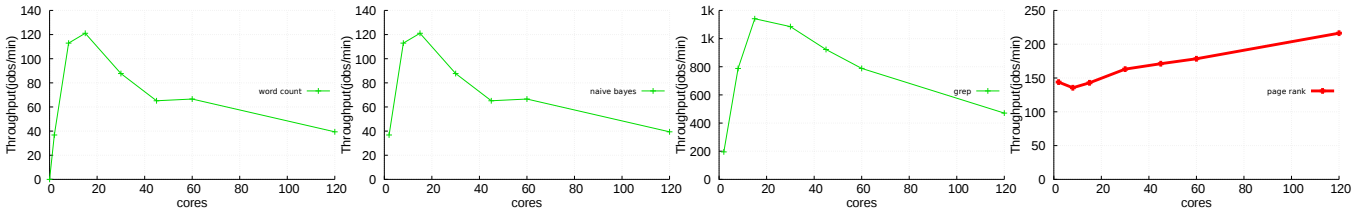
5. RELATED WORK

5.1 Apache Spark

5.2 Manycore Partitioning

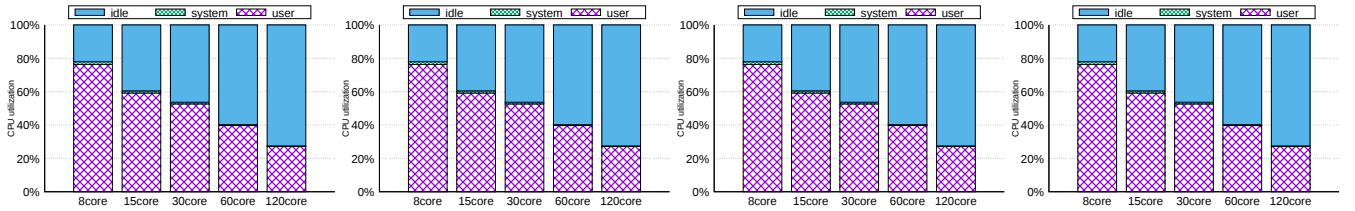
5.3 Manycore Scalability

6. CONCLUSION



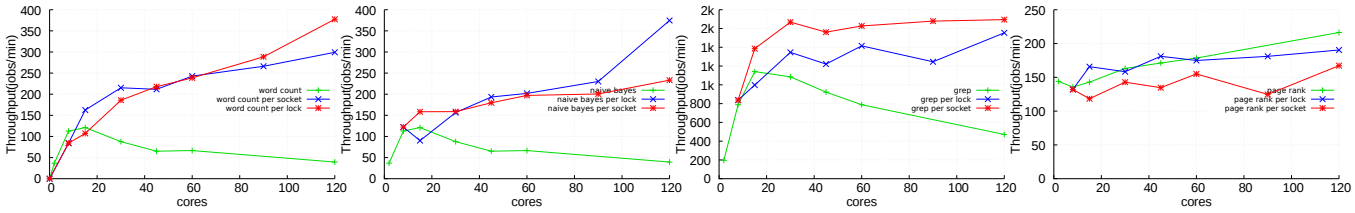
(a) Word Count (b) Naive Basian (c) Grep (d) Pagerank

Figure 1: CPU utilization on 120 core.



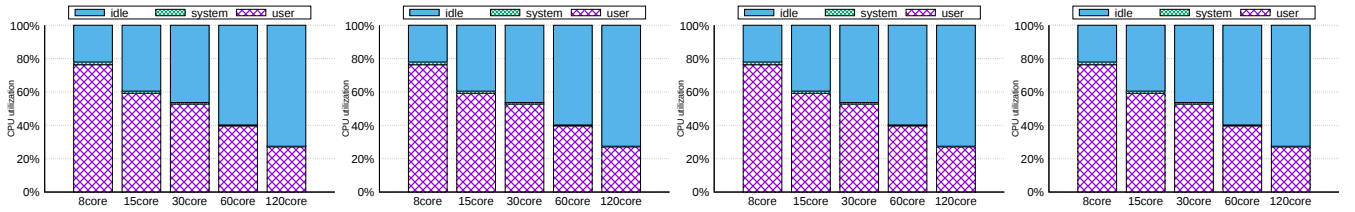
(a) Word Count (b) Naive Basian (c) Grep (d) Pagerank

Figure 2: CPU utilization on 120 core.



(a) Exim - 120core (b) Lmbench - 120core (c) AIM7 - 120core (d) Lmbench - 120core

Figure 3: CPU utilization on 120 core.



(a) Word Count (b) Naive Basian (c) Grep (d) Pagerank

Figure 4: CPU utilization on 120 core.