Comparison of the HPC and Big Data Java Libraries Spark, PCJ and APGAS

Jonas Posner, Lukas Reitz, and Claudia Fohry

jonas.posner@uni-kassel.de

UNIKASSEL
VERSITÄT

November 16, 2018

Motivation

HPC	Big Data
• Performance	• Fault tolerance
• Algorithmic flexibility	• Productivity
• C/C++ with MPI and/or OpenMP	• JVM-based, Java and Scala

Motivation

- A unified environment would be welcome
- Assumption: bring Java to HPC may help to bridge the gap
- We compare the performance and productivity of
  - Spark
  - PCJ
  - APGAS

Spark

- Implements MapReduce
- Maintains data in memory
- Primary data abstraction: Resilient Distributed Datasets
- Data are automatically distributed
- Transformations produce new RDDs (e.g. `map`)
- Actions extract results from RDDs (e.g. `reduce`)

PCJ

- Implements the Partitioned Global Address Space model
- Place = memory partition + computational resources
- Each place is realized by a single JVM
- Single Program Multiple Data
- All workers carry out the same code
- Different code paths
- Data can be exchanged between workers

APGAS

- Implements an asynchronous variant of the PGAS model
- Computations are encapsulated in asynchronous tasks
- Tasks have to be mapped manually to places
- **Intra**-place load balancing is performed automatically

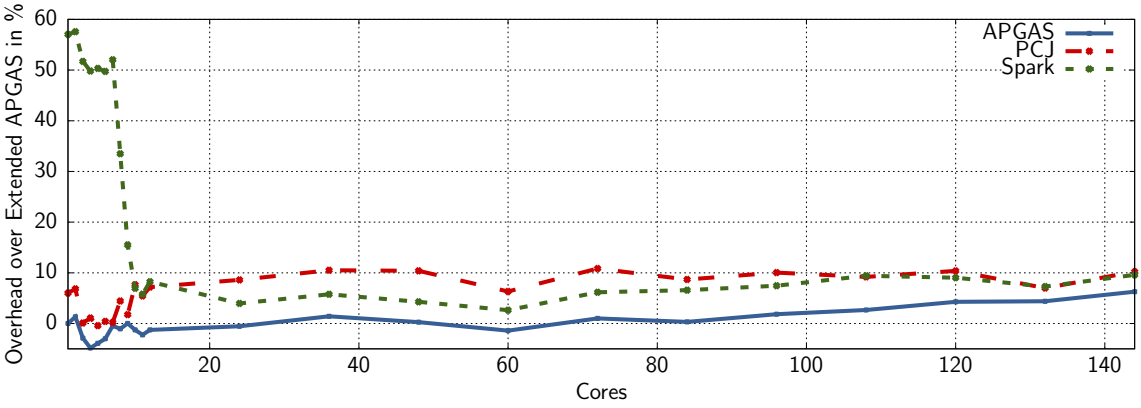
Extended APGAS: locality flexible tasks

- Own extension adds locality flexible tasks
- These tasks may run equally well on any resource of the system
- Task results can be easily stored and reduced
- Intra- **and** inter-place load balancing is performed automatically

What we did

- Investigated the productivity, ease of use, and performance characteristics of those libraries
- Used three benchmarks:
  - WordCount, Pi, UTS
- Evaluated:
  - **performance** with strong scaling up to 144 cores
  - **productivity** using metrics and subjective impressions

π: Overhead over APGAS with locality flexible tasks



π: Productivity

Lines of code

Spark	PCJ	APGAS	Ex APGAS
29	67	36	31

Number of different library constructs used

Spark	PCJ	APGAS	Ex APGAS
7	12	6	7

Conclusions

Overall, the extended APGAS library

- was intuitive to use
- required the lowest number of constructs
- needed only a few more lines of code than Spark
- showed the best performance

$\rightarrow$ seems to have the potential to be a unified environment for both HPC and Big Data

Thank you for your attention!

Please feel free to ask questions