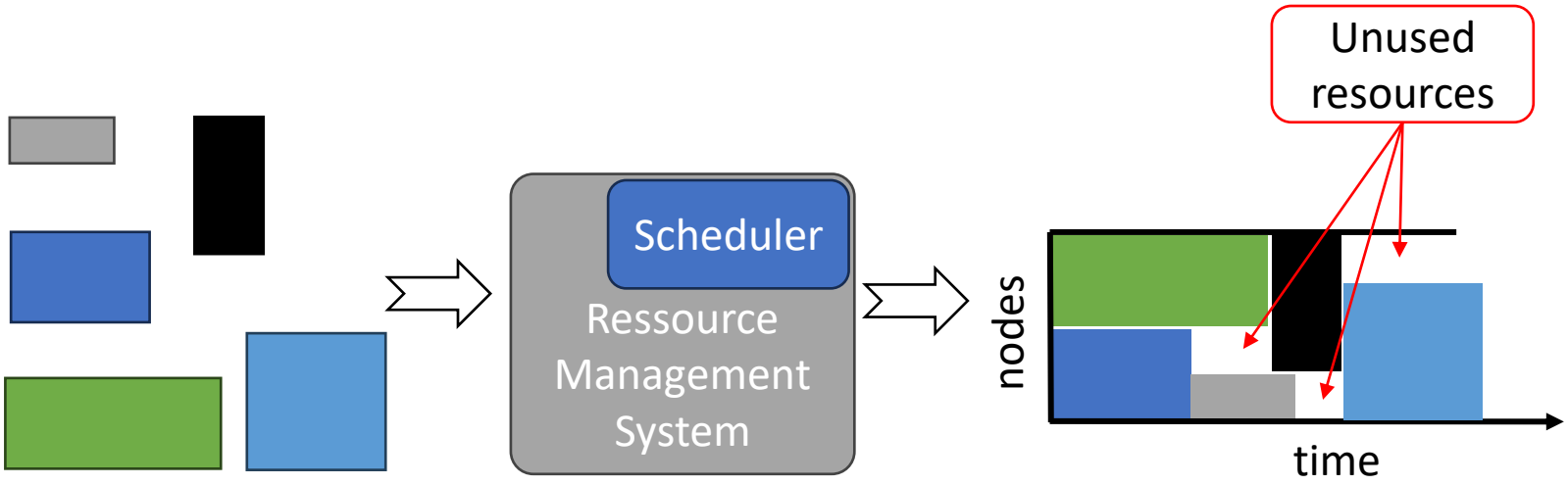
Malleable APGAS Programs and their support in Batch Job Scheduler

Patrick Finnerty[1], Leo Takaoka[1], Takuma Kanzaki[1], Jonas Posner[2]

1. Kobe University, Kobe, Japan
2. University of Kassel, Kassel, Germany

Parallel Job Scheduling

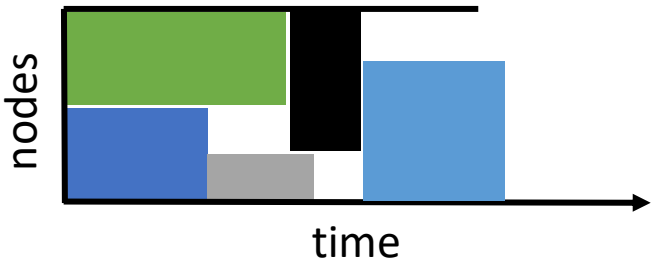


Jobs submitted by users

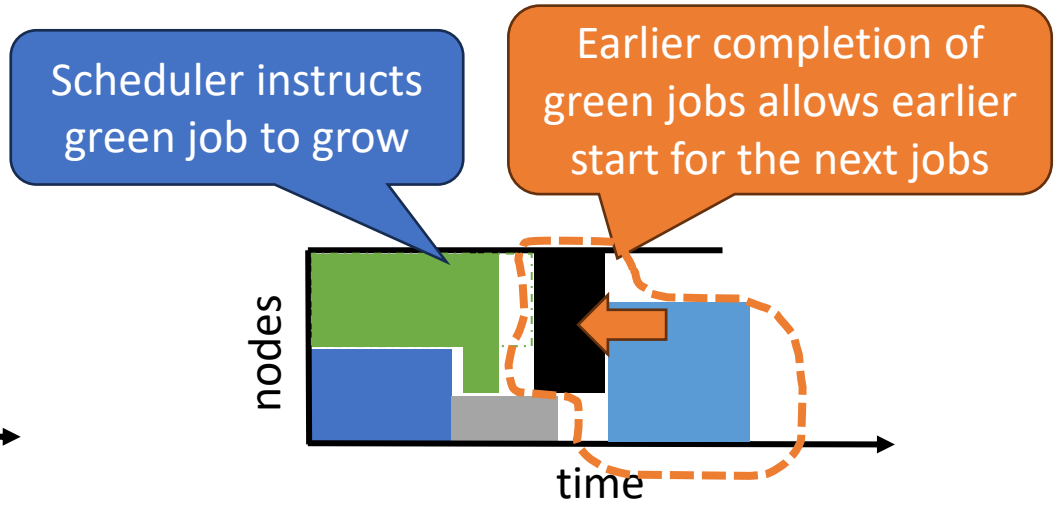
Possible schedule on
parallel machines

Malleable Job Potential

- Malleable Job?
  → can change nb of nodes used (grow/shrink) during execution following scheduler instruction



Possible schedule on parallel machines with rigid jobs only

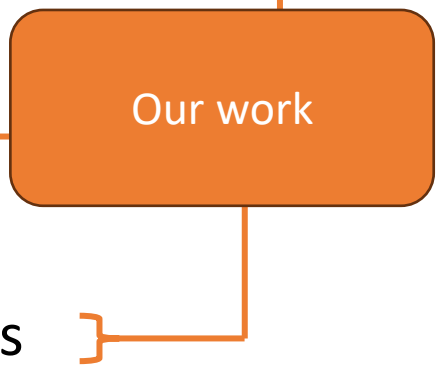


Possible schedule on parallel machines with some malleable jobs

- Benefits?
  → More effective use of resources
  → Increased throughput and/or reduced energy consumption

Hurdles

- Support in RMS (Slurm, Torque, Open PBS, …)
  - Interactions between program/RMS necessary
  - Some experimental implementations
- Support in libraries and Programming models
  - MPI w/ Checkpoint/Restart
  - Charm++
  - (A)PGAS languages
- Compatible programs
  - Requires effort to convert/create applications

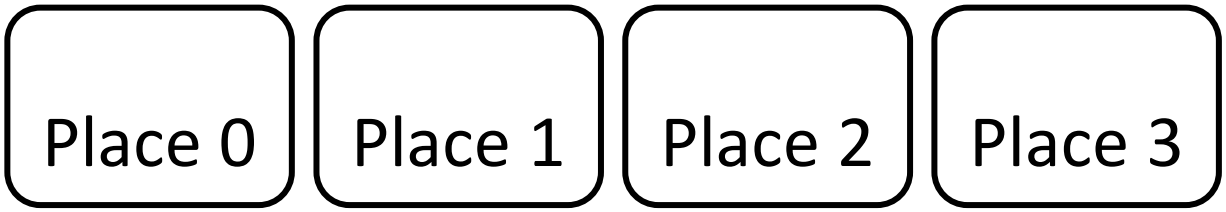


Contributions

- Malleable implementation of APGAS for Java
  - Simple abstractions for programmers
  - Modularity for adaptation to future schedulers
- Refactoring of an AMT work-stealing scheme
  - Simplified through our abstractions
  - Can respond to scheduler directives
- Demonstrate the benefits of malleability on a Beowulf cluster with a custom scheduler

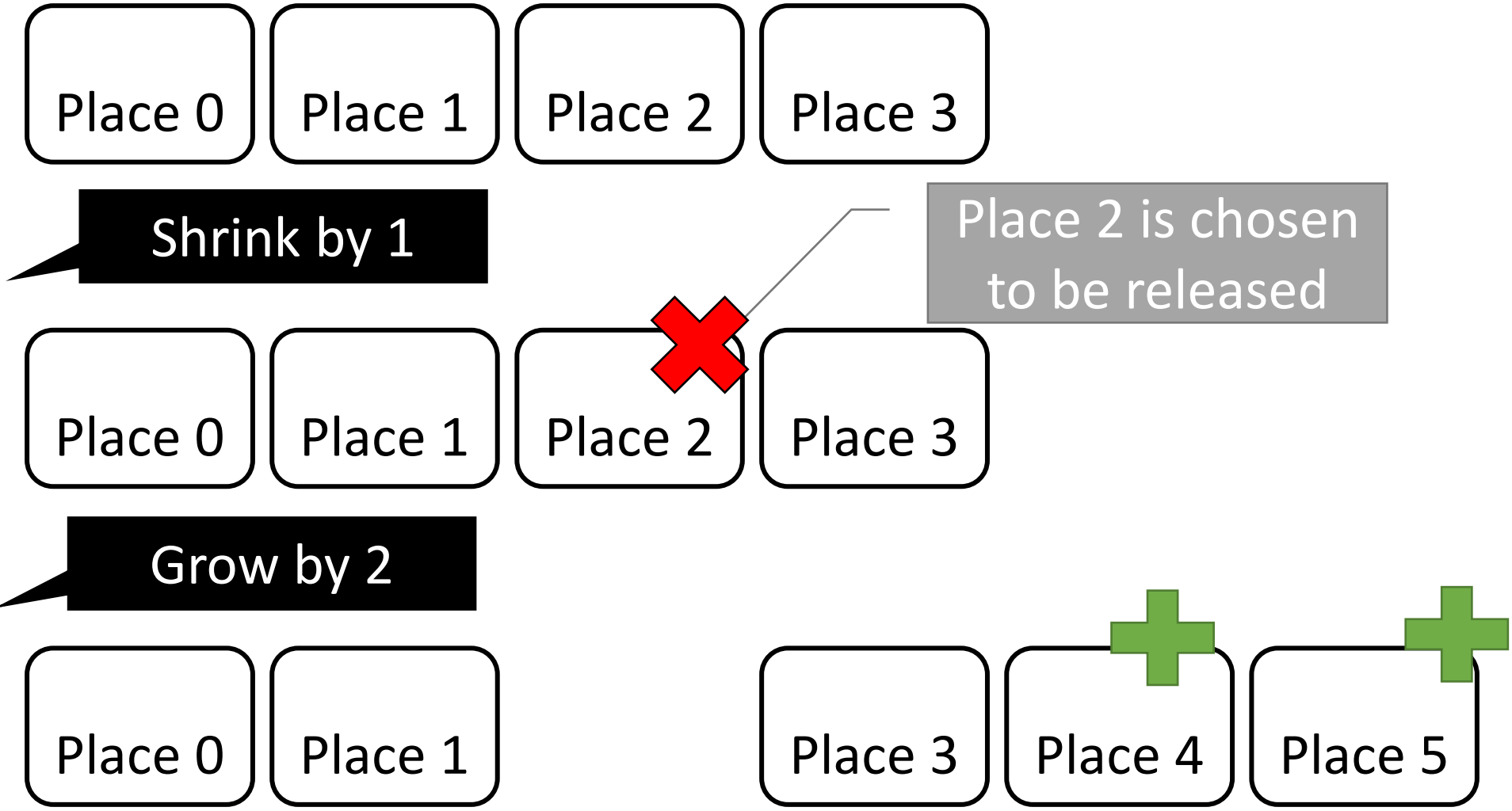
A word about (A)PGAS

- Partitioned Global Address Space
  - The memory is partitioned between processes
  - That memory can be accessed from remote processes
- Asynchronous PGAS
  - Can spawn and control termination of tasks on the processes
- In X10 and APGAS for Java:
  - “Place,” finish at async

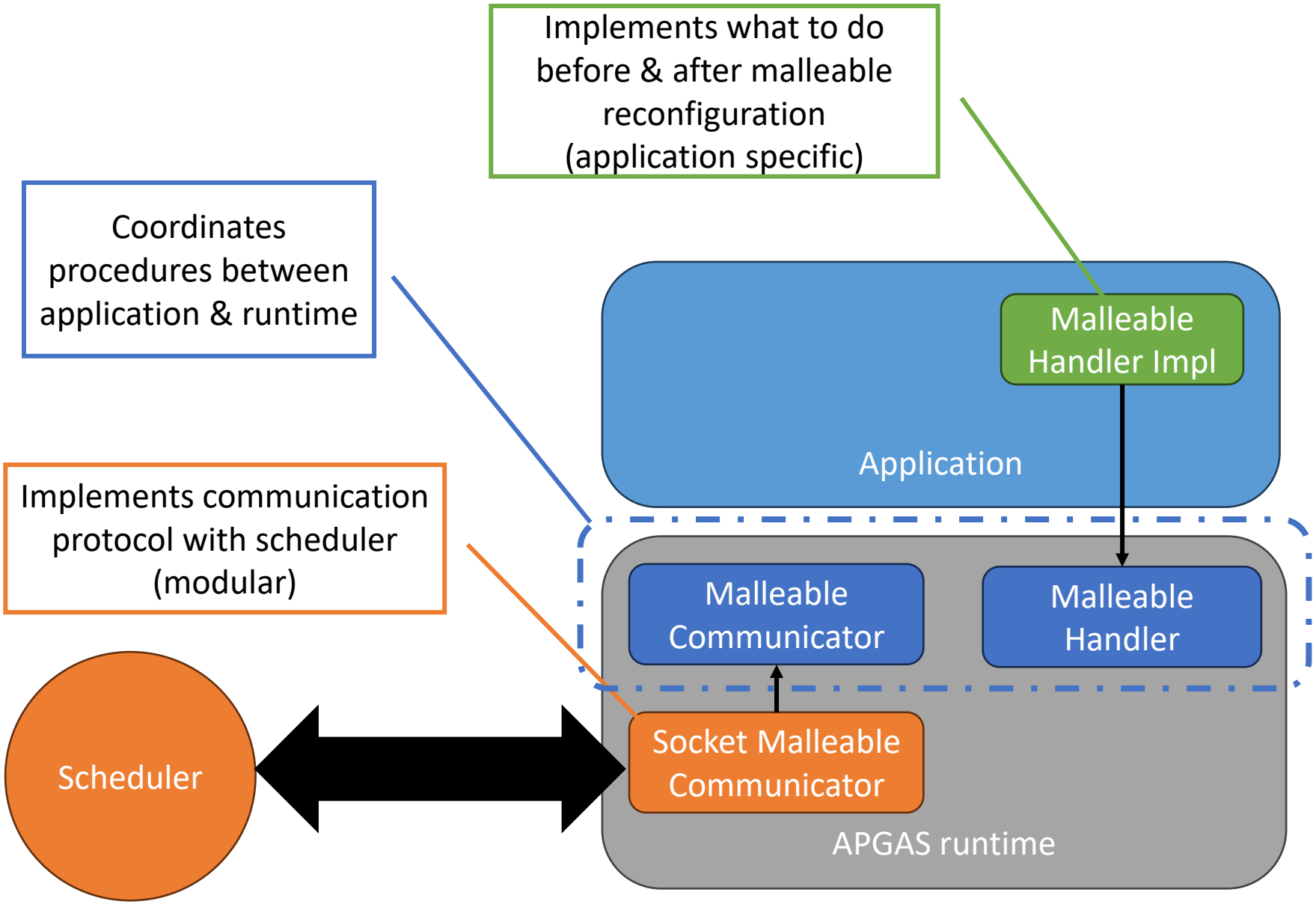


Malleable APGAS?

- Add and remove Places!

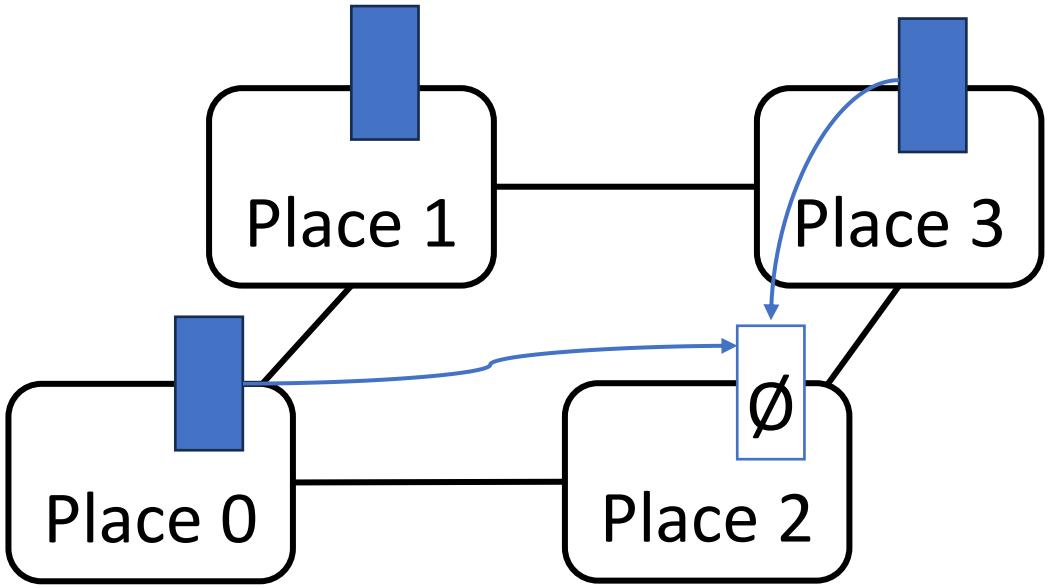


Malleable APGAS architecture



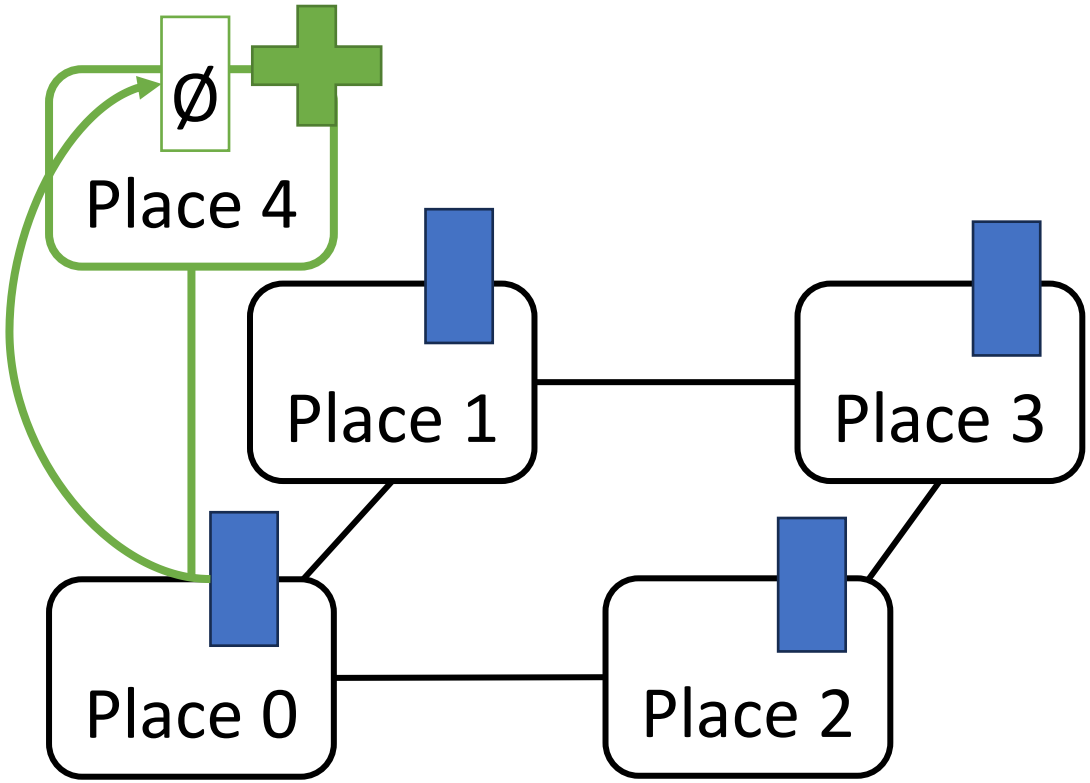
Application example

- Lifeline-based Global Load Balancer
  - Each place has some tasks
  - When a place runs out of tasks, steal through preferential channels, the “lifelines”



Application example

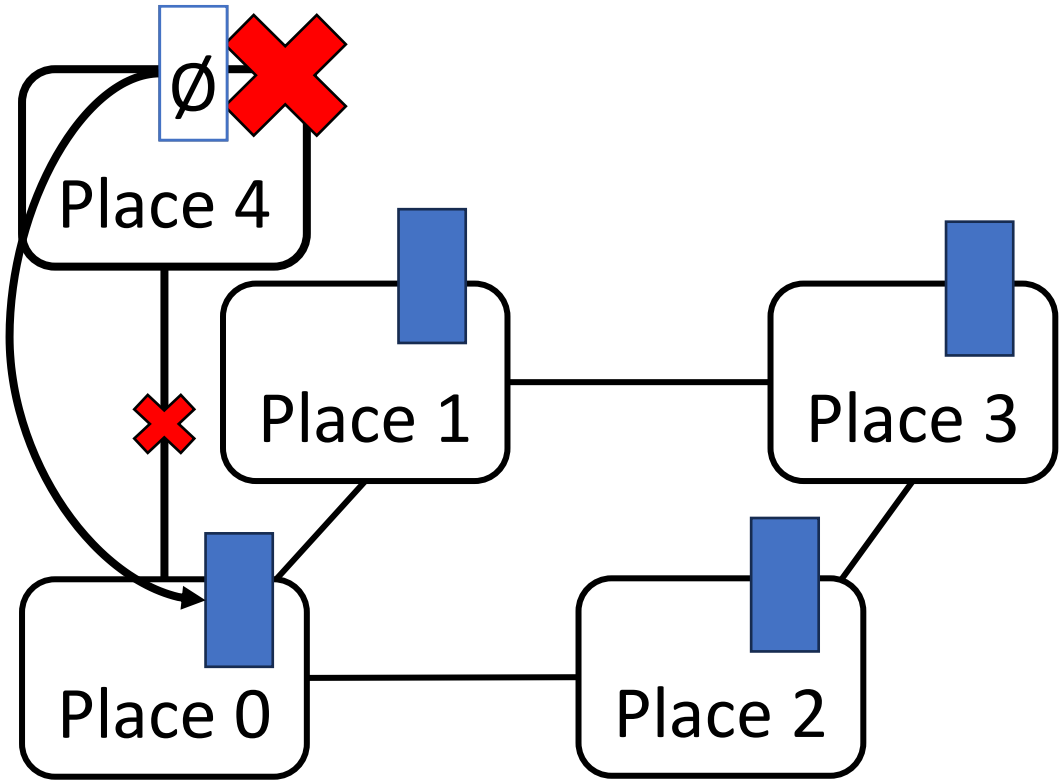
- Grow order
  - Before: do nothing
  - After: integrate the new place(s) into the lifeline network, they steal some work and start working



Note:
Computation continues while new places are added to the runtime!

Application example

- Shrink order
  - Before: Disconnect place(s) from lifeline network, relocate any work to remaining places
  - After: do nothing

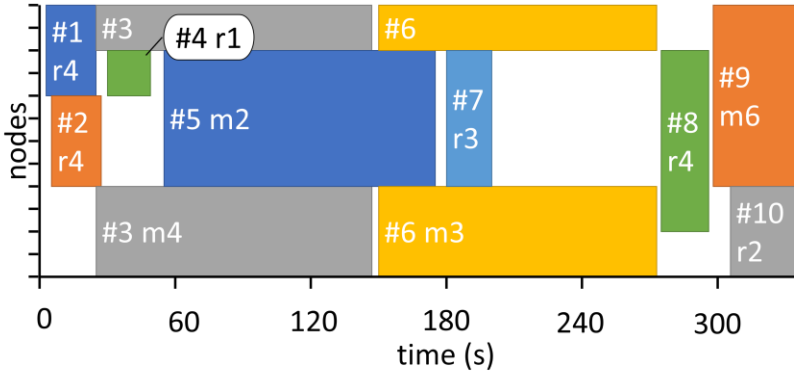


Note:
Computation continues while places are removed from the runtime!

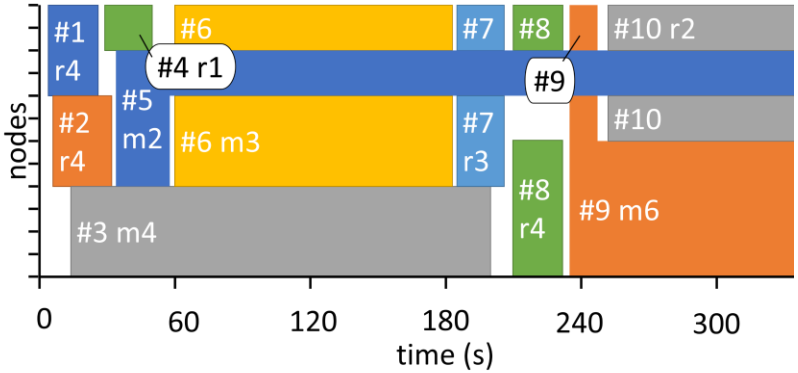
Evaluation - environment

- Beowulf cluster
  - 12 nodes
  - Password-less SSH authentication
- Simplistic malleable job scheduler
  - First-Come First-Served
  - If possible, shrink jobs to allow the next one to start
  - Otherwise, grow running malleable jobs
- 30-job batch
  - Some MPI, some APGAS
  - Malleable programs have a min/max nb of nodes
  - Compare “100% fixed” with “half malleable/half fixed” batch

Evaluation - results



100% fixed jobs



Half fixed,
half malleable

Workload:	100% rigid	Half malleable half rigid
Makespan (m)	21.4	**18.2 (-15%)**
Avg cluster utilization	72.3%	83.4% (+15%)
Avg wait time (m)	9.4	**7.9 (-15%)**
Avg exec time (m)	0.96	**1.40 (+45%)**
Avg response time (m)	10.36	**9.34 (-10%)**

Higher throughput!

Malleable jobs use fewer nodes for longer

Conclusion

- Presented a practical, extensible, malleable implementation of APGAS for Java
- Demonstrated performance benefits on our small Beowulf, even with a simplistic scheduling strategy
- Perspectives
  - Support for evolving jobs? Other applications?
    - Change in resource allotment initiated by the program
  - Evaluation on larger clusters?
    - Possible within a certain degree …
    - Scheduling algorithms best studied through simulation

Conclusion

- All of our source code is freely available:
  - Malleable APGAS runtime
    https://github.com/projectwagomu/apgas
  - Malleable lifeline-based Global Load Balancer
    https://github.com/projectwagomu/lifelineglb
  - Custom scheduler
    https://github.com/projectwagomu/FIFO-Malleable-Job-Scheduler

Thank you for listening!