CRIU in HPC

Adrian Reber
areber@redhat.com

August 20, 2015

Background

- Computer Simulation used for product development
  - Faster development cycles
  - More predictable quality
- Goal: Increase model complexity/granularity
- $\rightarrow$ Increase number of nodes
- New challenges for system management
- $\rightarrow$ New approaches necessary

Virtualization in HPC

- Every CPU cycle counts
- Hypervisor overhead for the CPU is low, but is exists
  - $\rightarrow$ para-virtualization
  - $\rightarrow$ container based virtualization
- Overhead in combination with I/O even higher ($\rightarrow$ emulation)
- Migration difficult in combination with out-of-band communication

Process Migration

- Remove overhead by directly migrating processes
- Handle out-of-band communication in software layer

Migration Methods - Postcopy

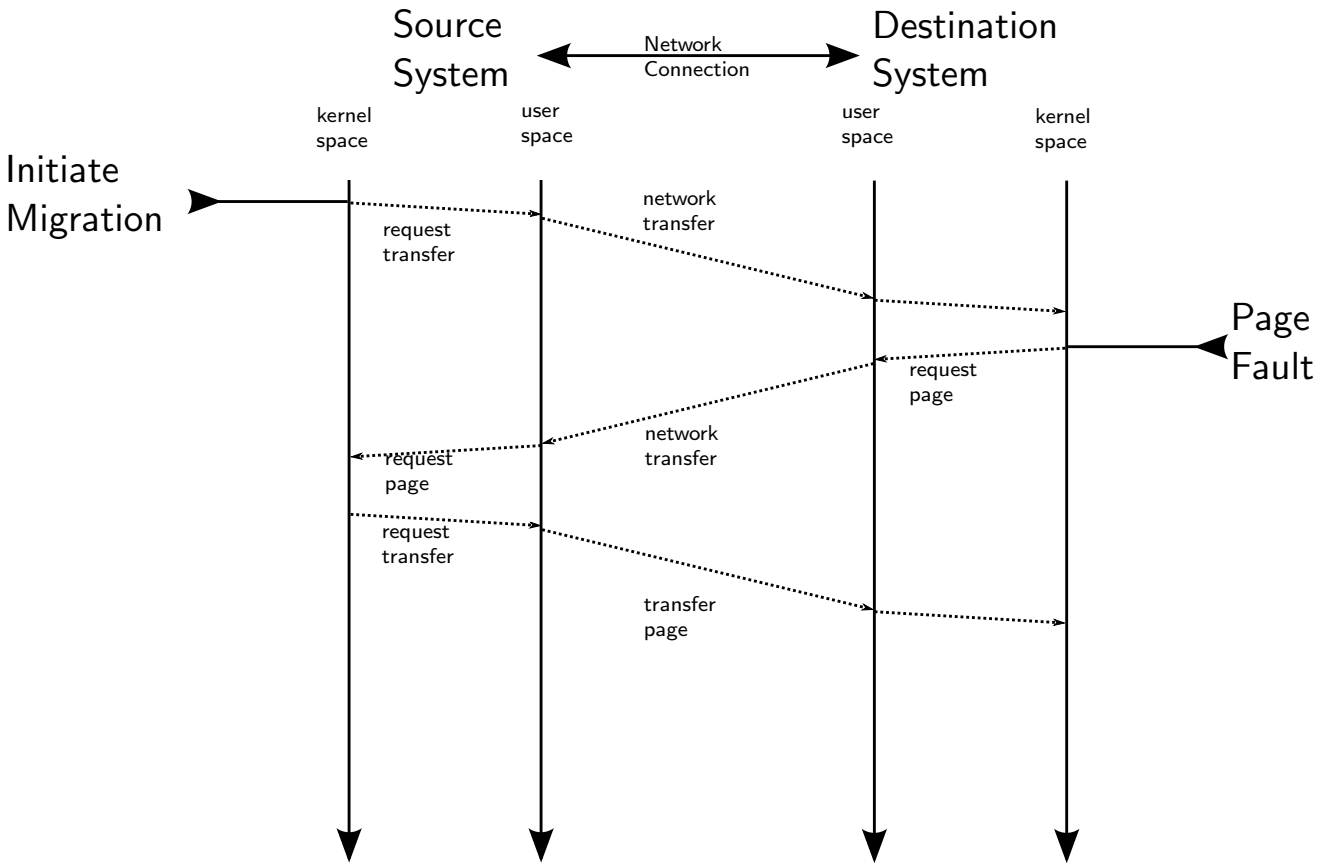

Figure: Postcopy Migration

Migration Methods - Checkpoint/Restart

- There are many existing Checkpoint/Restart implementations
- Low impact on the operating system

Checkpoint/Restart Requirements

- As transparent as possible
- Not too invasive
- Upstream inclusion

Berkeley Lab Checkpoint Restart - BLCR

- Used in many fault tolerance projects
- Almost transparent (pre-loading or re-compilation)
- Implemented as external kernel module

Distributed MultiThreaded Checkpointing - DMTCP

- Implemented in user-space
- Requires pre-loading
- Intercepts many system calls

Kernel-Space-Based

- Developed with the goal of upstream inclusion
- Developed in cooperation with the Linux kernel community
- Transparent solution without pre-loading or re-compilation
- Solution to complicated $\rightarrow$ needs changes in every Linux kernel subsystem

User-Space-Based - CRIU

- Developed with the goal of upstream inclusion
- Transparent solution without pre-loading or re-compilation
- Uses existing interfaces as much as possible
- Most functionality resides in user-space

Parallel Process Migration

- Based on Open MPI
  - Open license and community
  - Flexible fault tolerance framework

Parallel Process Migration - Start

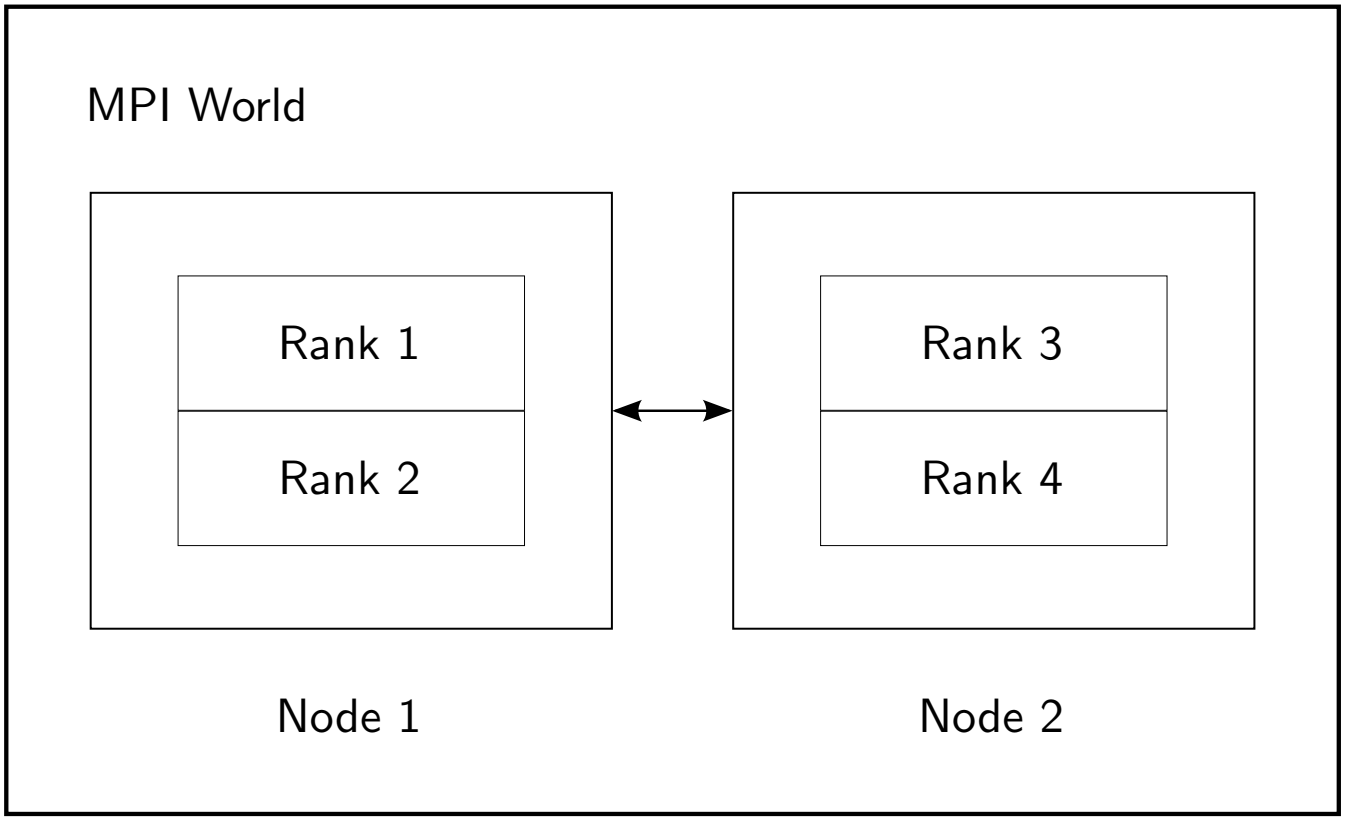

Figure: Parallel Process Migration - Start

Parallel Process Migration - Complete Node

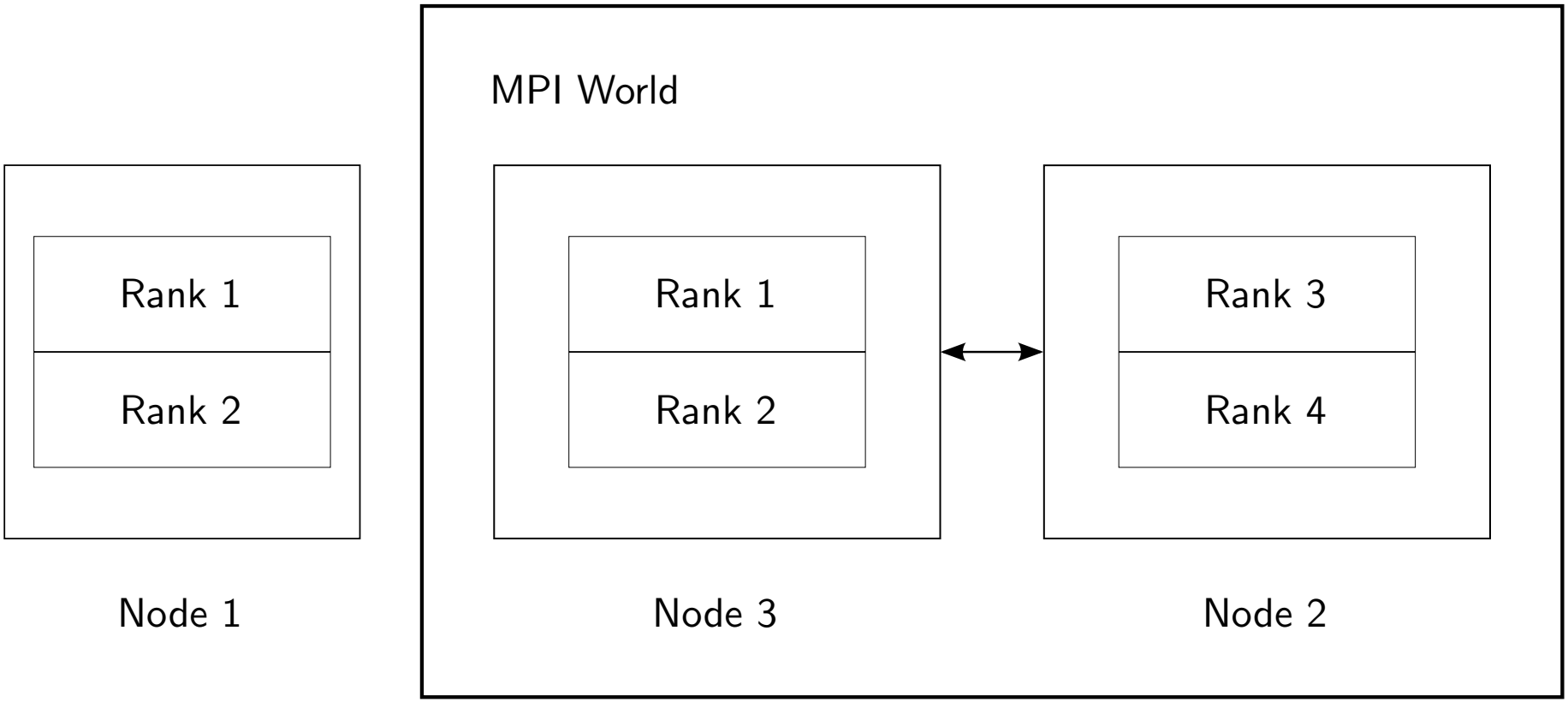

Figure: Parallel Process Migration - Complete Node

Parallel Process Migration - Load Balancing

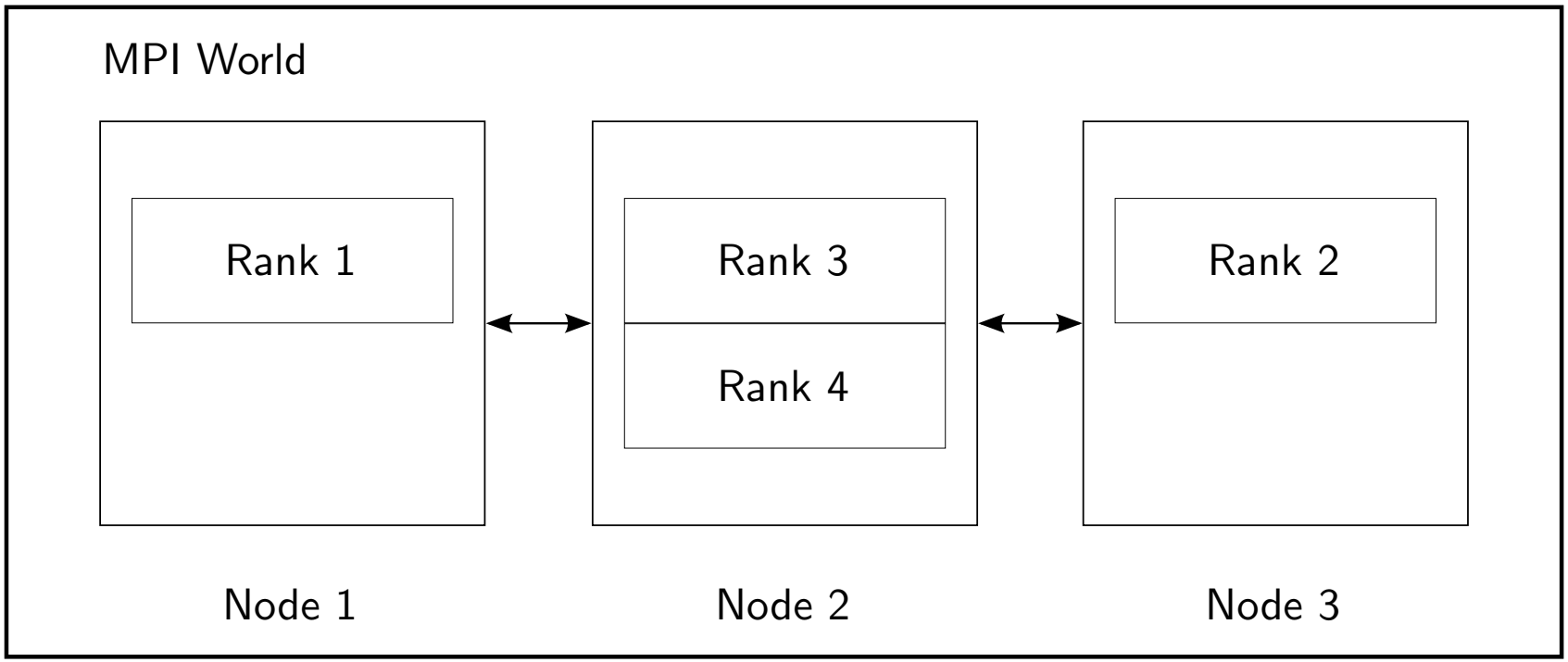

Figure: Parallel Process Migration - Load Balancing

Conclusion

- Transparent process migration works (based on CRIU)
- Open MPI extended to support CRIU
- Parallel process migration work in progress