MARS

Multicore Application Runtime System

July 11, 2008

Geoff Levand

<geoffrey.levand@am.sony.com>

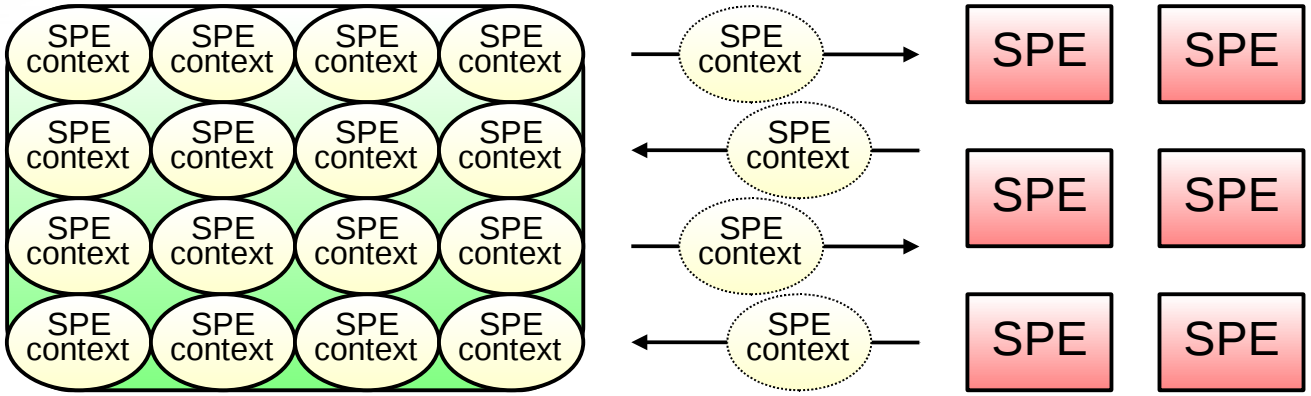
Linux Kernel SPE Context Scheduler

- Best performance when:

  (# of SPE contexts) <= (# of physical SPEs available)

- High context switching overhead when:

  (# of SPE contexts) > (# of physical SPEs available)



What is MARS?

- MPU-centric runtime environment for multicore architectures
- MPU-side scheduling of workloads
- APIs to manage user programs that run on MPUs

MARS Terminology

- **MARS** – Multicore Application Runtime System
- **Host** – host processor (**PPE**)
- **MPU** – micro-processing unit (**SPE**)
- **Host storage** – shared memory space (**main memory**)
- **MPU storage** – MPU local memory space (**local store**)
- **Workload** – a generic unit of process(es) scheduled for execution on the MPU (**SPE context**)

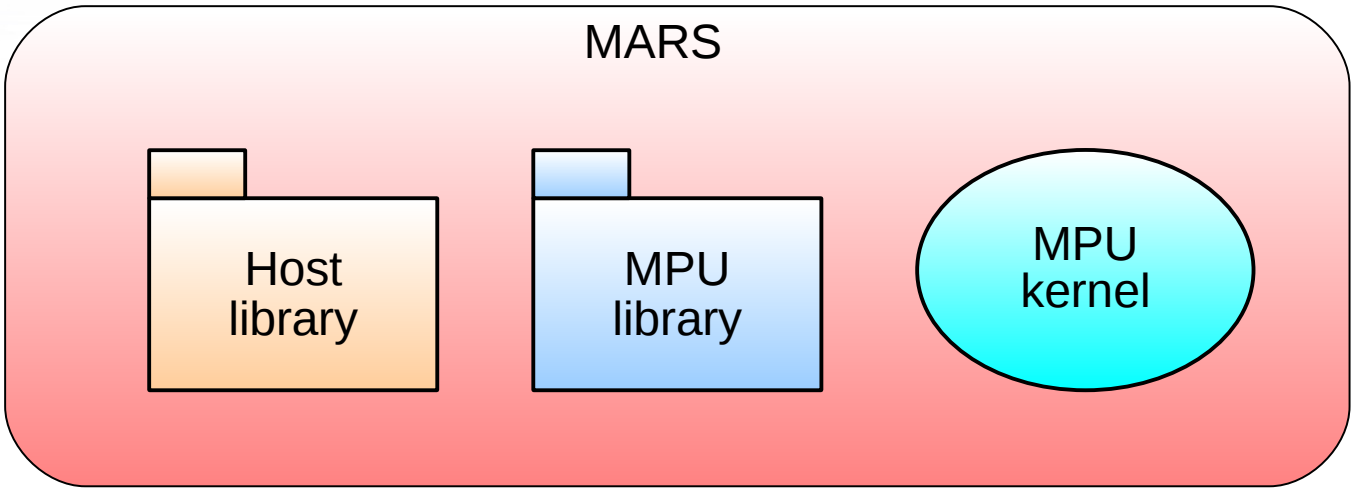
Why use MARS?

- Lightweight context switching
- Performance advantage over libspe when:

  (# **of workloads) > (# of physical MPUs available**)
- Minimizes runtime load of the host processor
- Synchronization objects call the scheduler

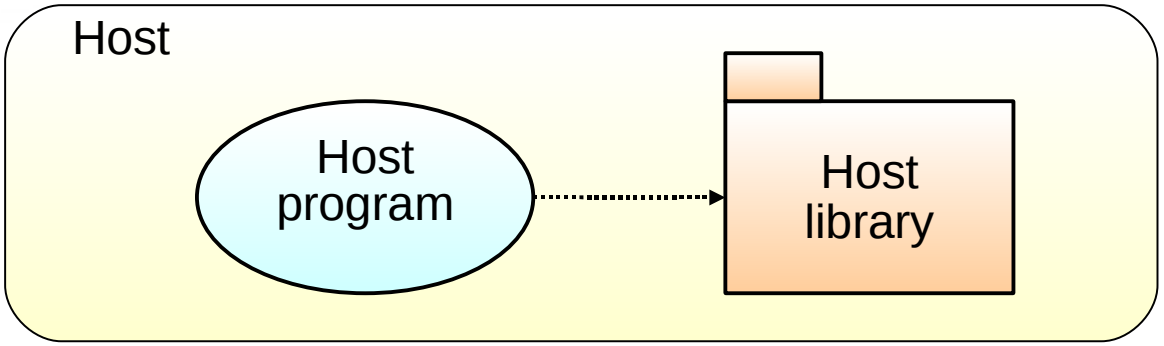
What does MARS provide?

- Host-side programming library
- MPU-side programming library
- MPU-side kernel



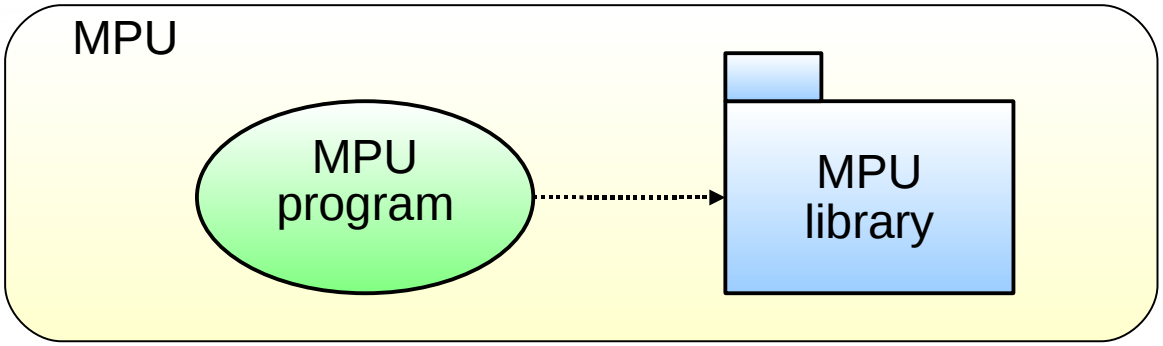
MARS Host Programming Library

- APIs to manage the MARS context
- APIs to initialize/schedule workloads for execution
- APIs to synchronize Host and MPU execution



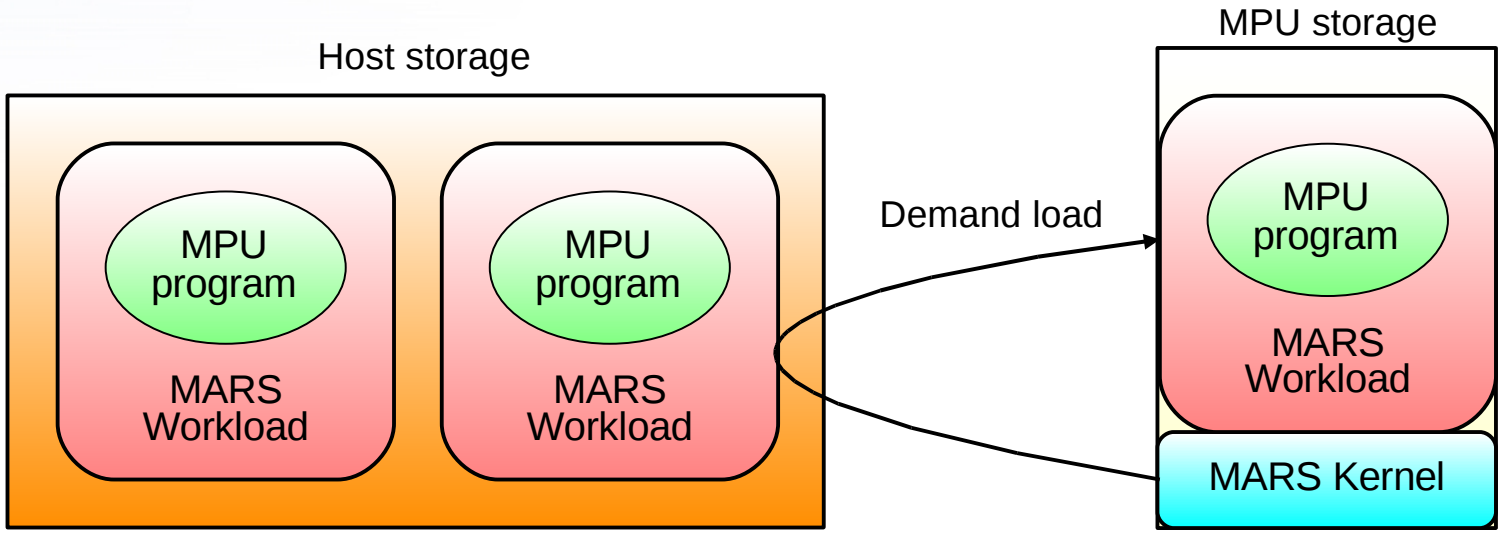
MARS MPU Programming Library

- APIs to manage MPU program state (ex. yield, exit, etc.)
- APIs to schedule initialized workloads
- APIs to synchronize Host and other MPUs

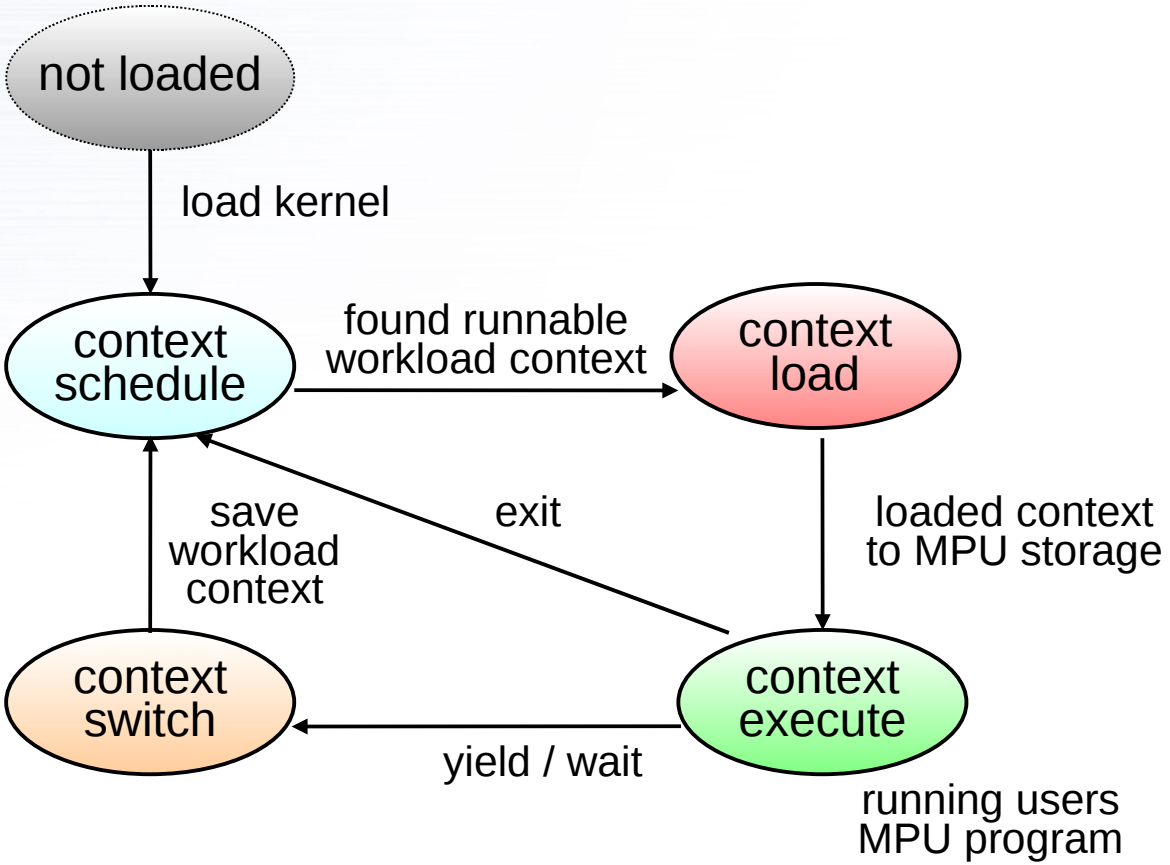


MARS MPU Kernel

- Resident in MPU storage throughout life of MARS context
- Demand loads workloads from main memory to MPU Memory
- Priority-based cooperative scheduling



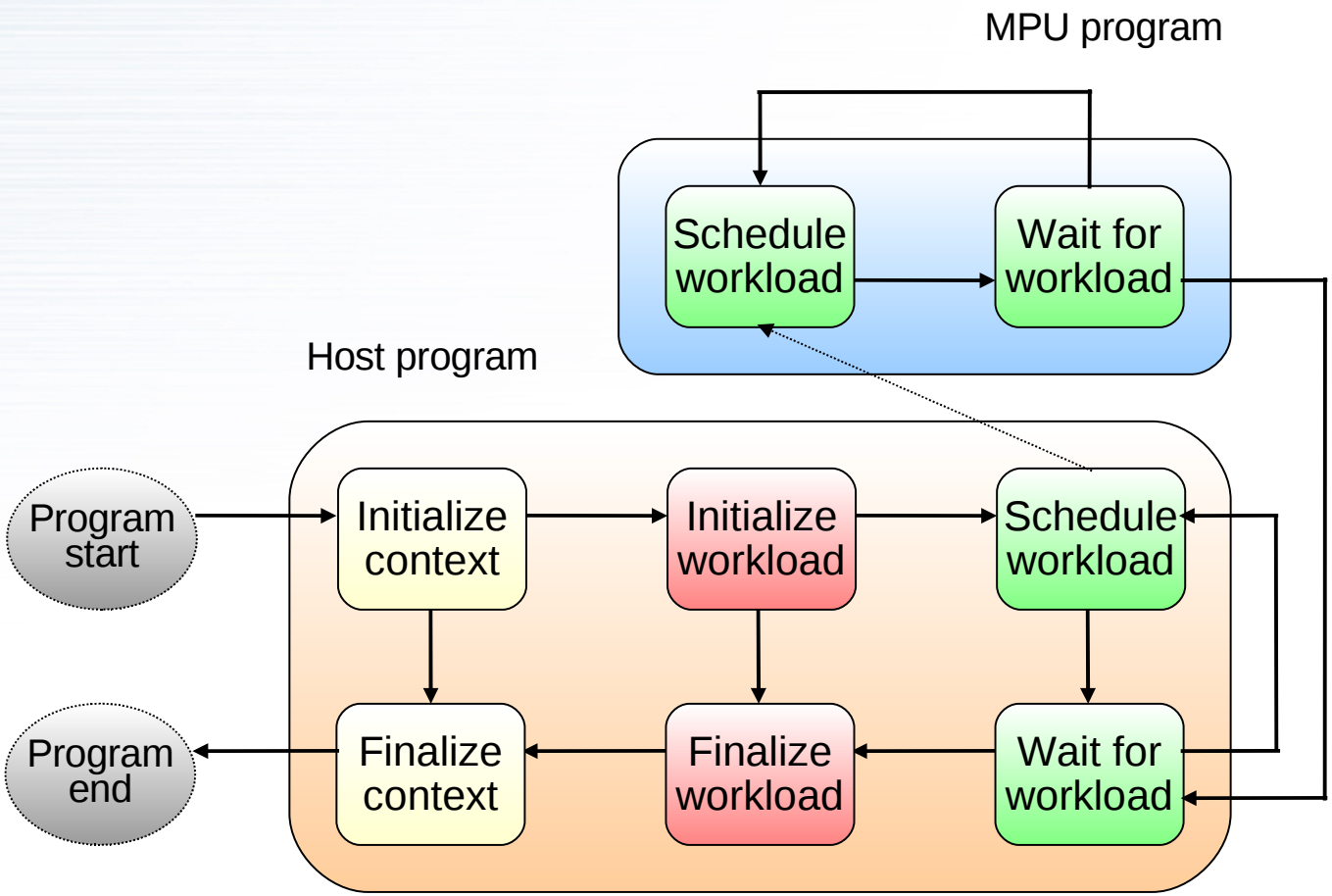
MARS MPU Execution State Diagram



MARS General Usage Flow

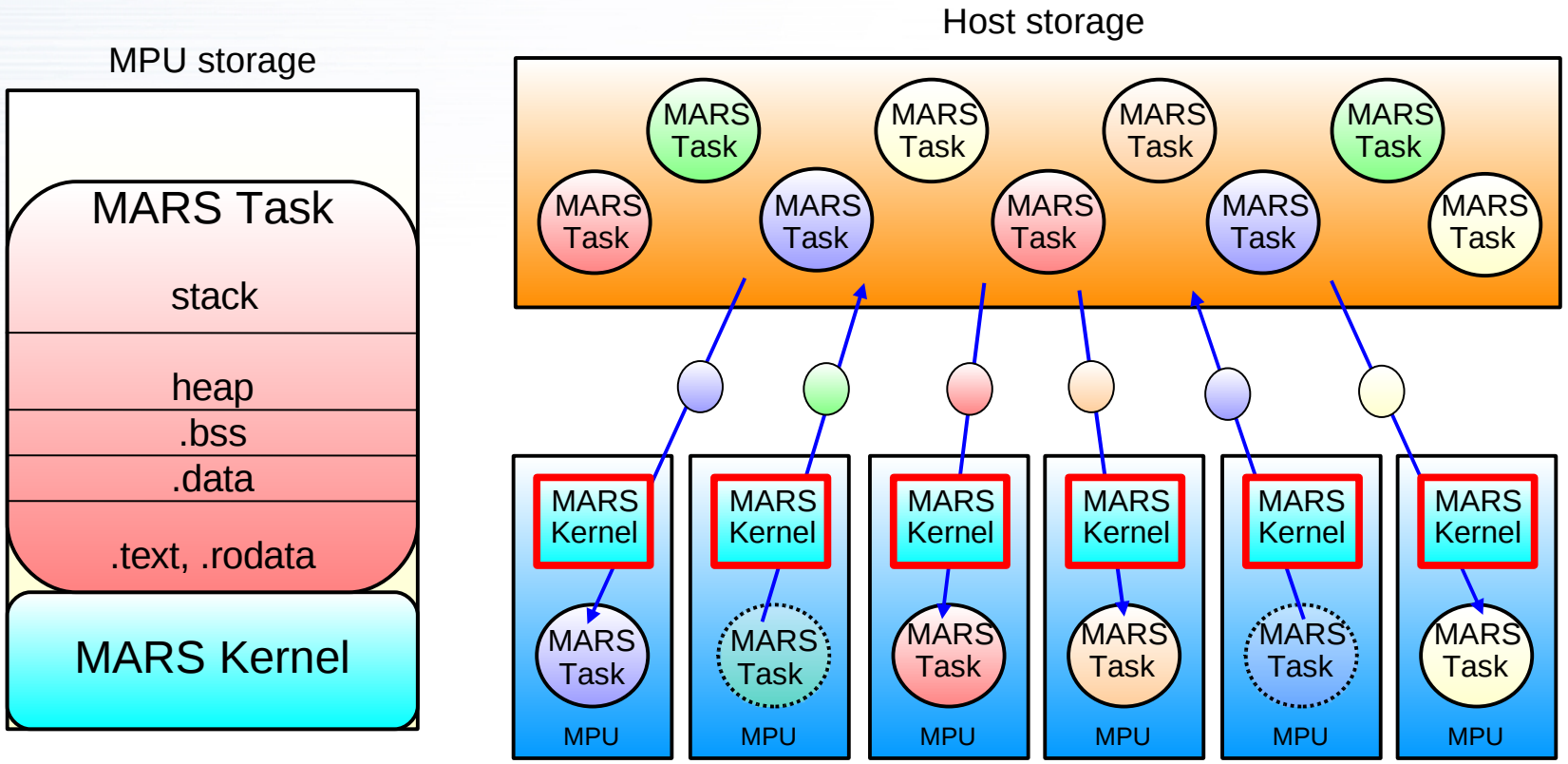
- Initialize MARS context
  - Prepares MARS kernel for each MPU
- Initialize workloads
  - Prepares workload contexts in shared workload queue
- Schedule workloads for execution
  - Workload scheduling can be requested from Host and MPU
- Wait for workload completion
  - Synchronous and asynchronous waiting provided
- Finalize MARS context

MARS General Usage Flow



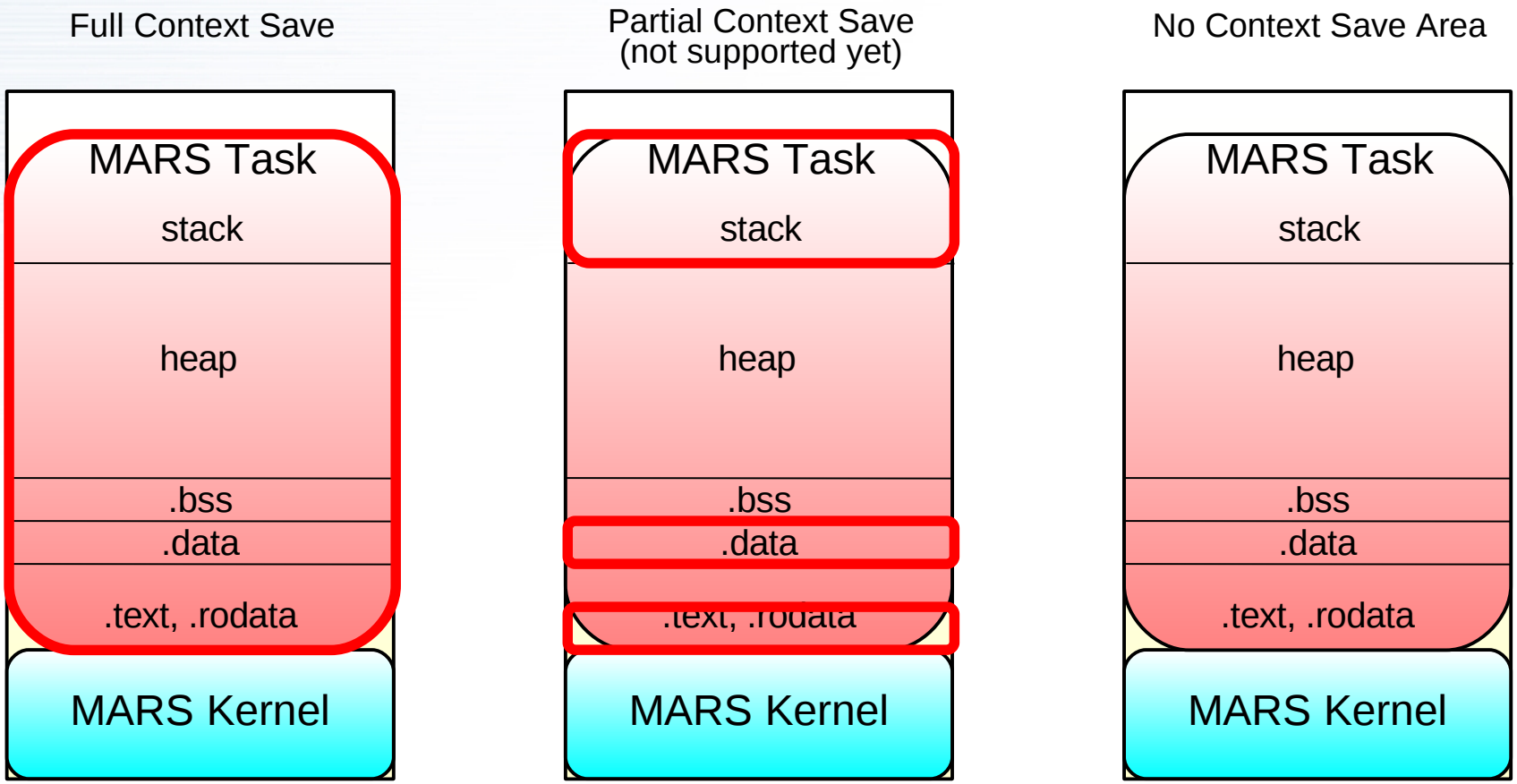
MARS Task Model

- Break up large processing into multiple smaller MARS Tasks
- Multi-task multiple unrelated tasks

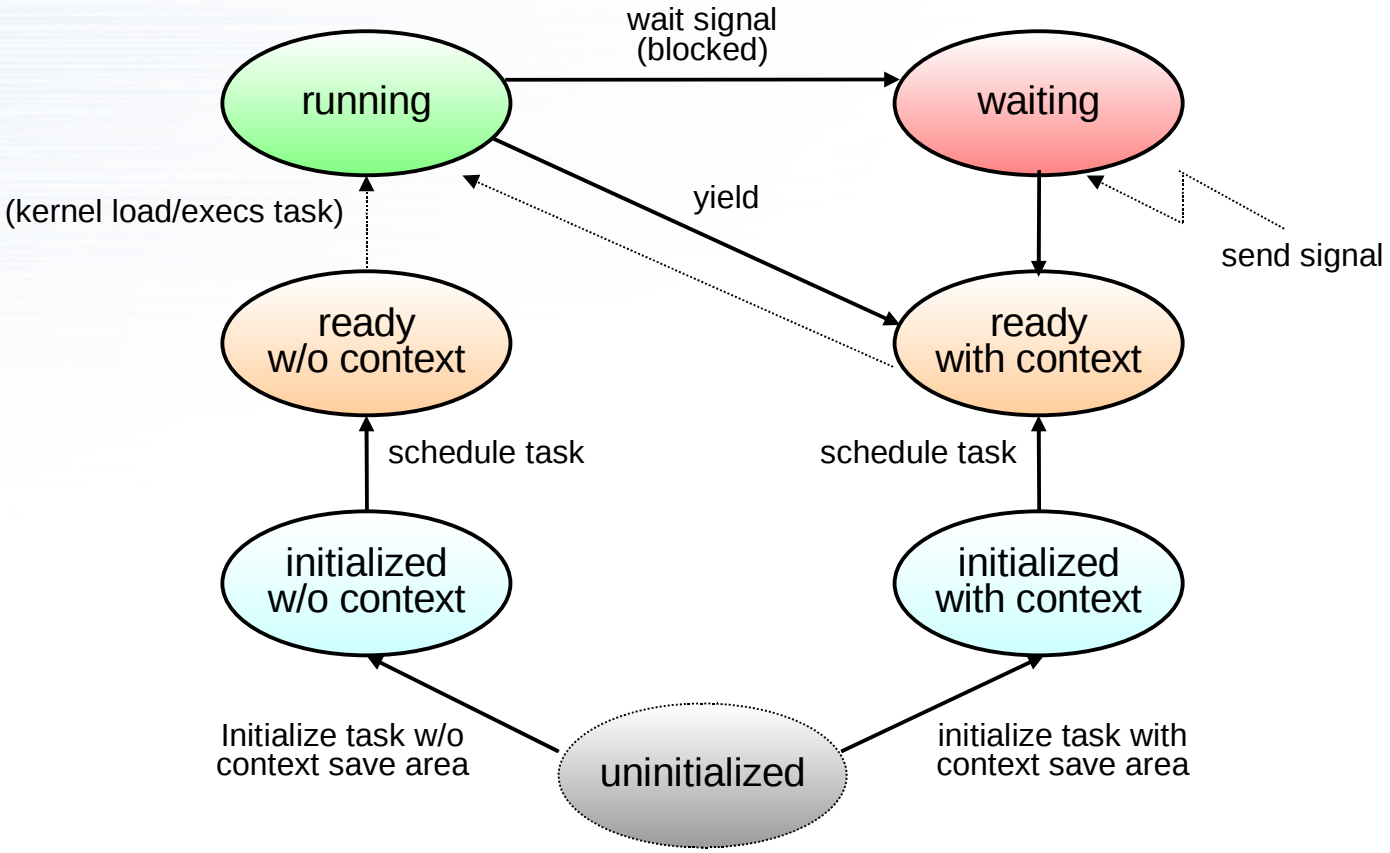


MARS Task Context Switch

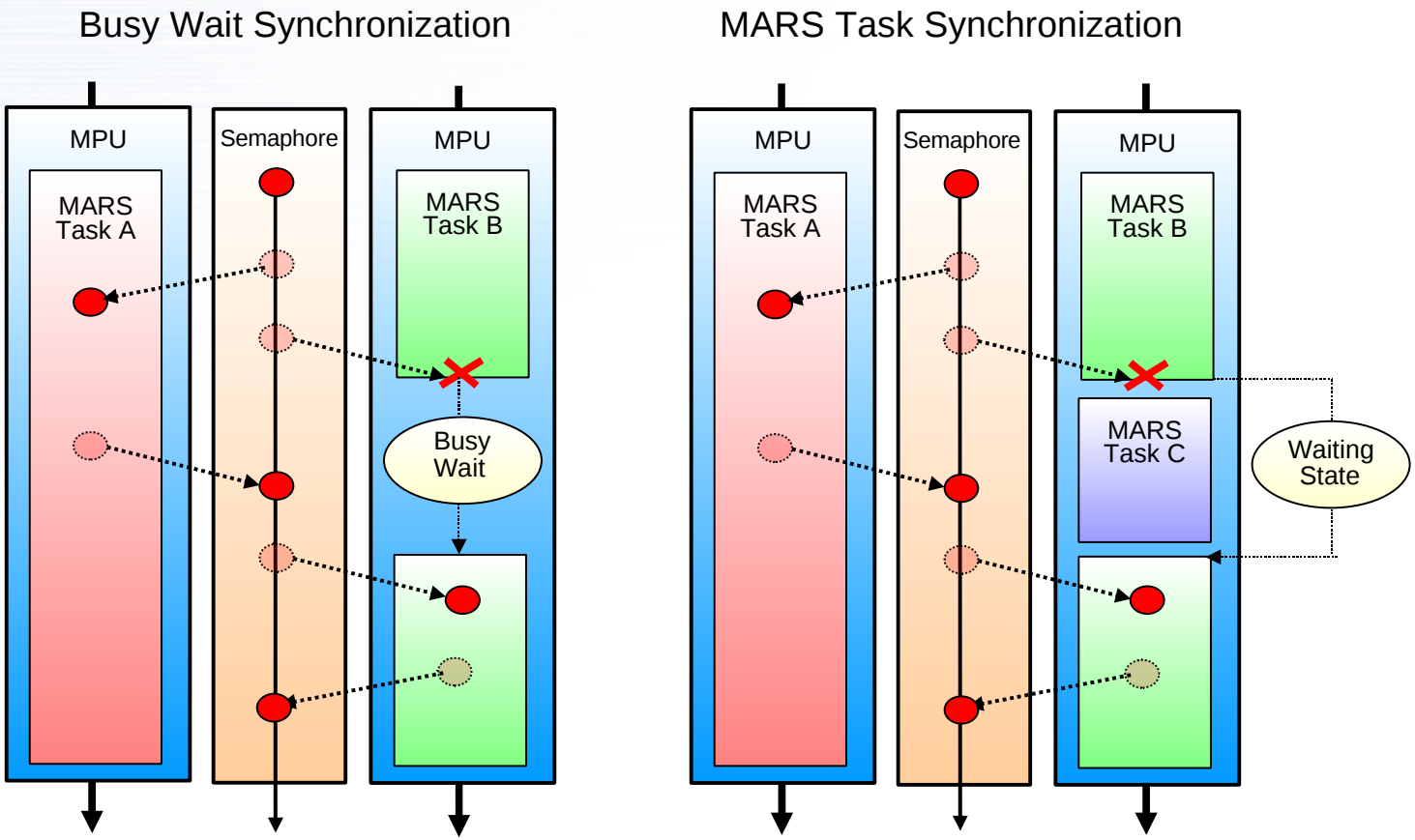
- Task context DMA'ed from main memory



MARS Task State Diagram



MARS Task Synchronization



Current Status of MARS

- Public release of prototype available
  - ftp://ftp.infradead.org/pub/Sony-PS3/mars/
- Support for task workload model
  - APIs for task management
  - APIs for task synchronization
    - event flag
    - barrier
    - queue
    - semaphore
    - signal

Future Plan for MARS

- Add support for other workload programming models
- MARS Task partial context save/restore
- Performance optimizations
- Feature improvements
- Test suite
- gdb debugger support

Thank You!