Charm++ : a very brief Introduction

Laxmikant (Sanjay) Kale
http://charm.cs.illinois.edu
Parallel Programming Laboratory
Department of Computer Science
University of Illinois at Urbana Champaign

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

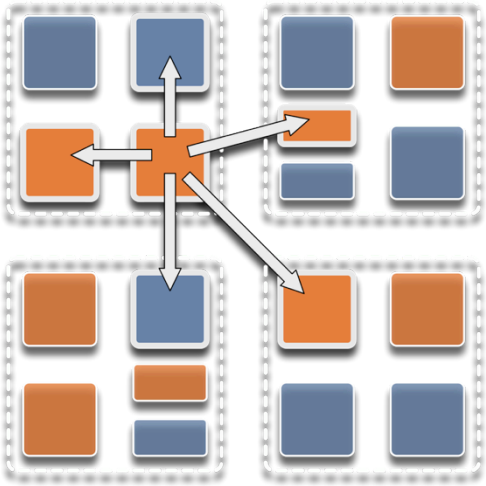
PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL UIUC

Exascale Challenges

- Main challenge: variability
  - Static/dynamic
  - Heterogeneity: processor types, process variation, ..
  - Power/Temperature/Energy
  - Component failure
- Exacerbated by strong scaling needs from apps
  - Why?
- To deal with these, we must seek
  - Not full automation
  - Not full burden on app-developers
  - But: a good division of labor between the system and app developers

Overdecomposition

- Decompose the work units & data units into many more pieces than execution units
  - Cores/Nodes/..
- Not so hard: we do decomposition anyway



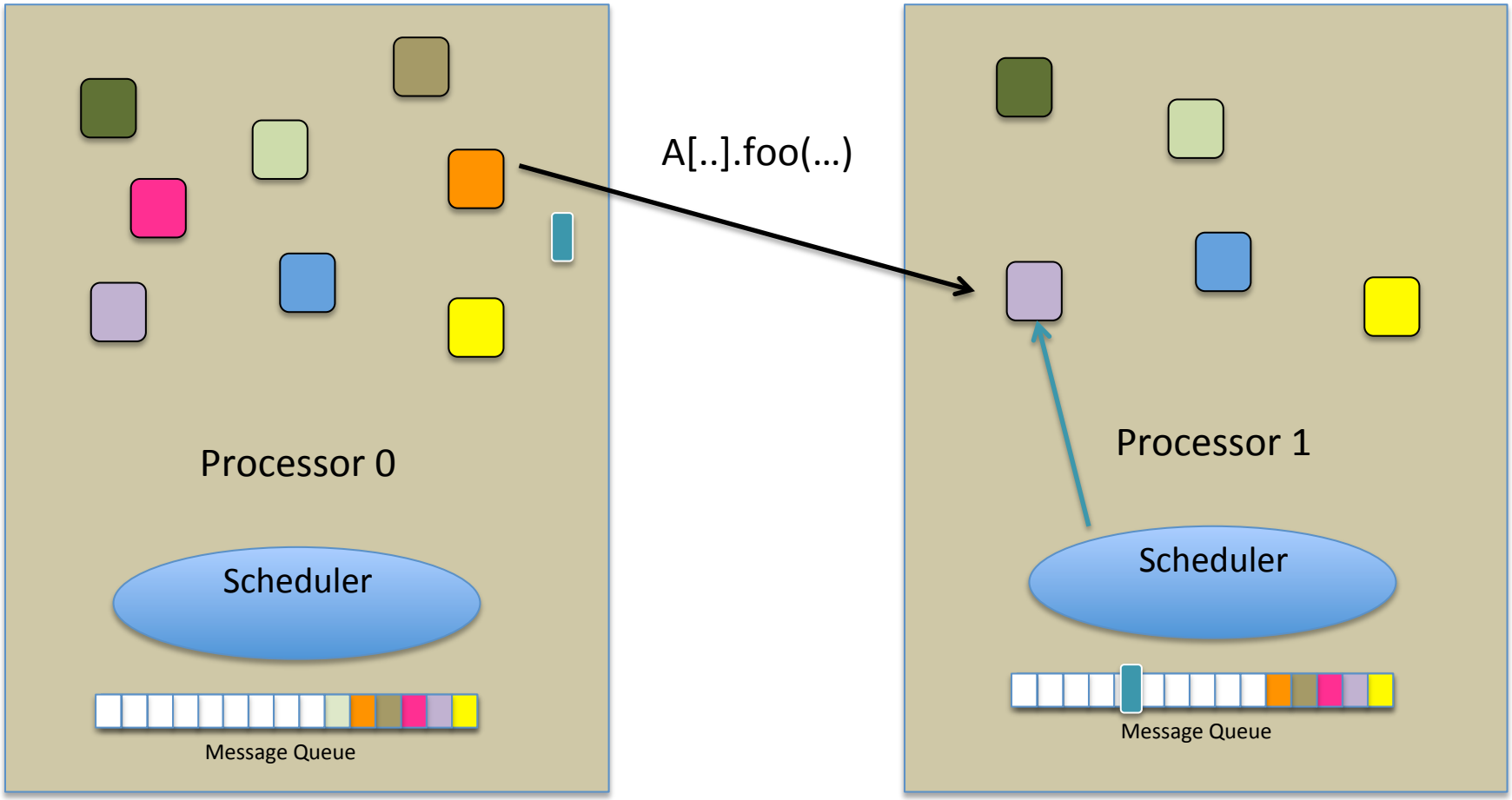
Migratability

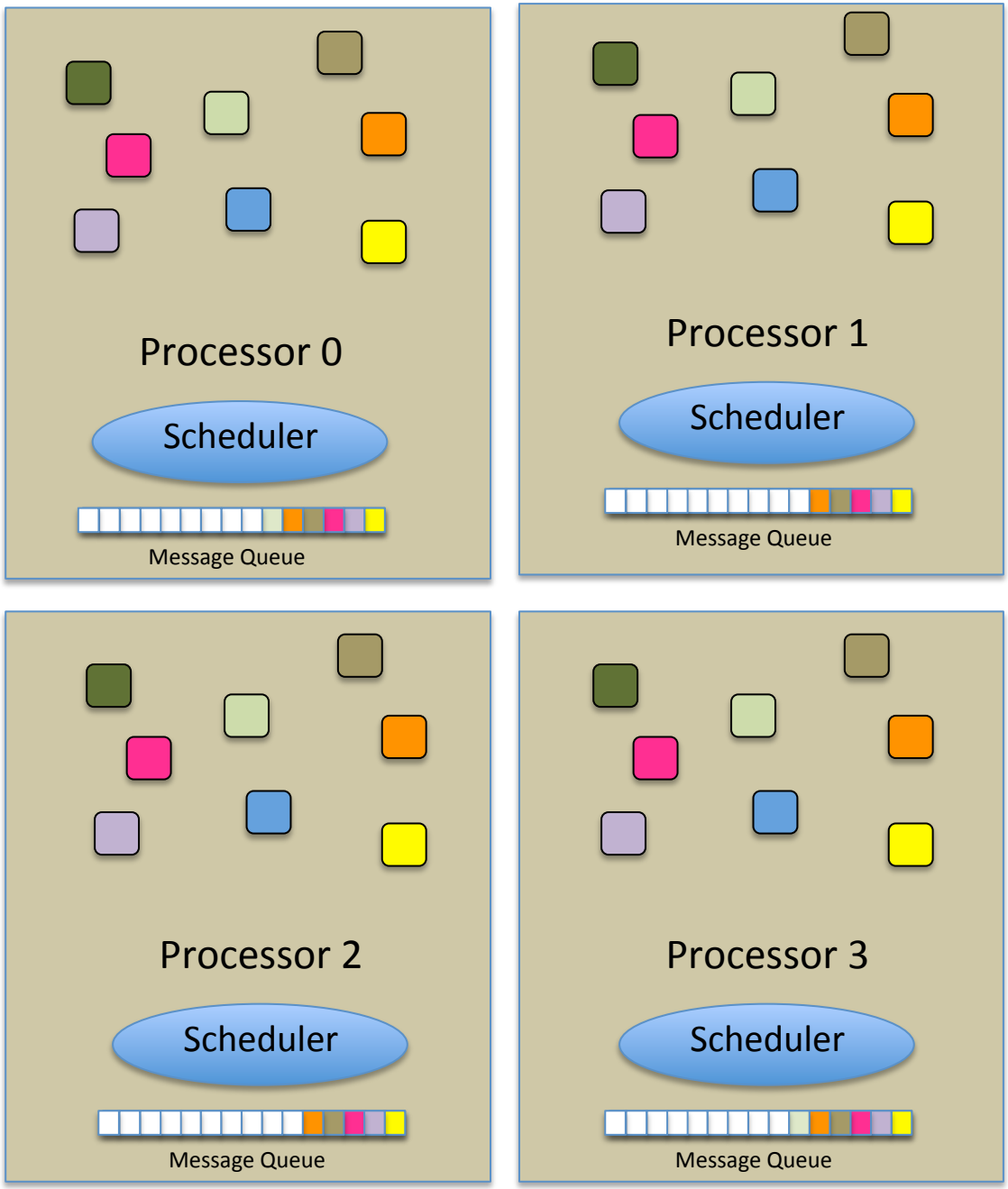
- Allow these work and data units to be migratable at runtime
  - i.e. the programmer or runtime, can move them
- Consequences for the app-developer
  - Communication must now be addressed to logical units with global names, not to physical processors
  - But this is a good thing
- Consequences for RTS
  - Must keep track of where each unit is
  - Naming and location management

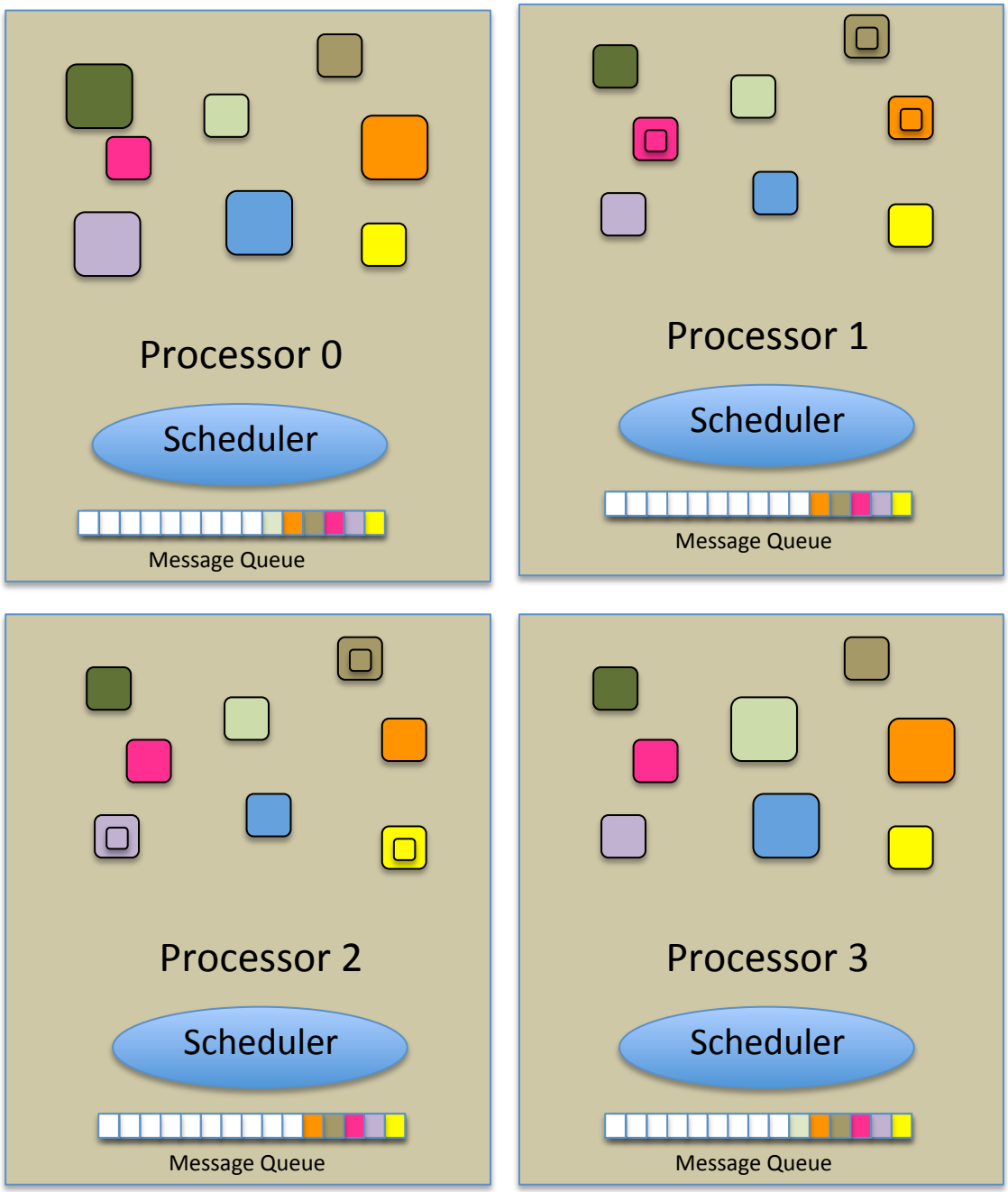
Asynchrony: Message-Driven Execution

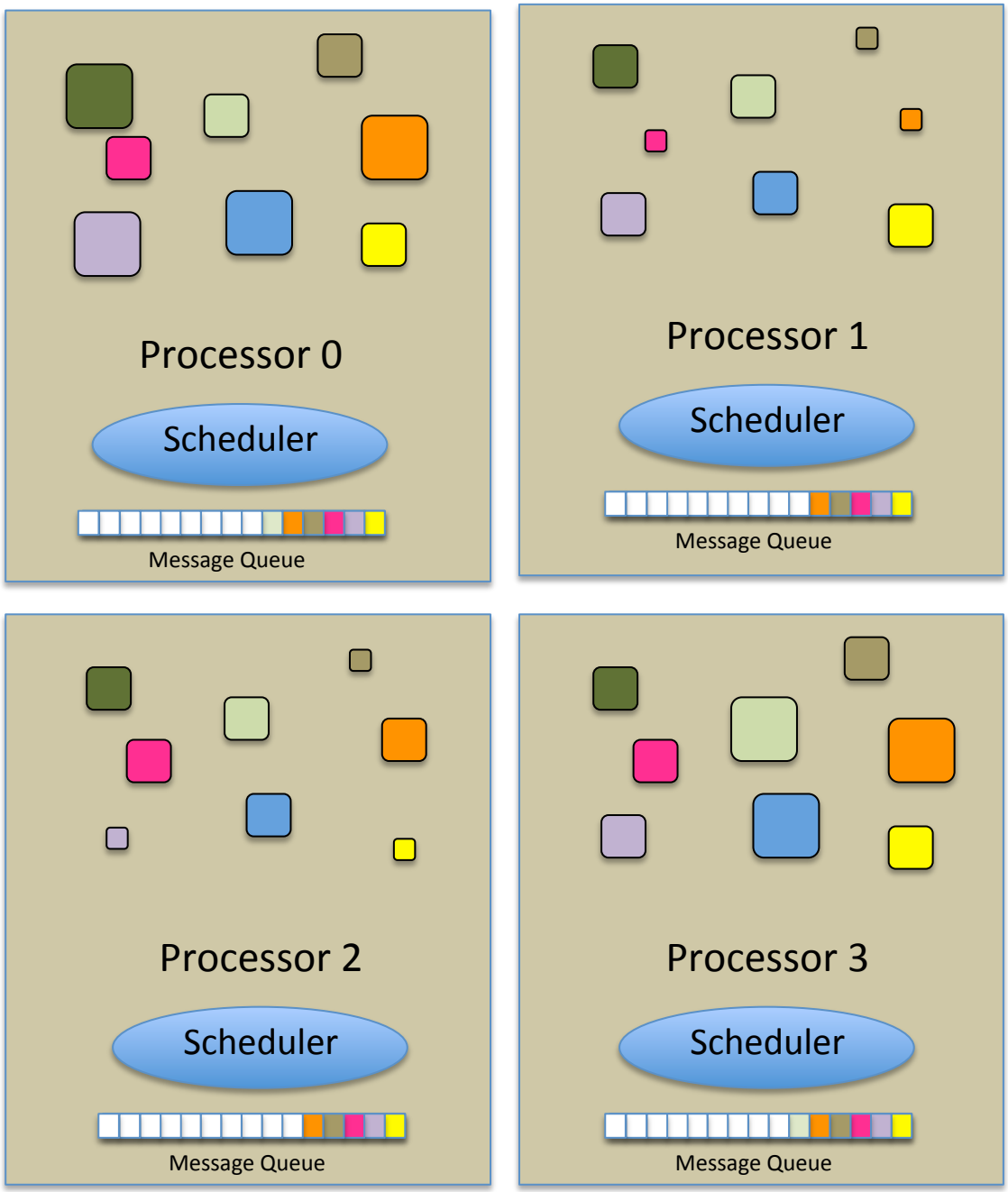
- Now:
  - You have multiple units on each processor
  - They address each other via logical names
- Need for scheduling:
  - What sequence should the work units execute in?
  - One answer: let the programmer sequence them
    - Seen in current codes, e.g. some AMR frameworks
  - Message-driven execution:
    - Let the work-unit that happens to have data ("message") available for it execute next
    - Let the RTS select among ready work units
    - Programmer should not specify what executes next, but can influence it via priorities

Message-driven Execution

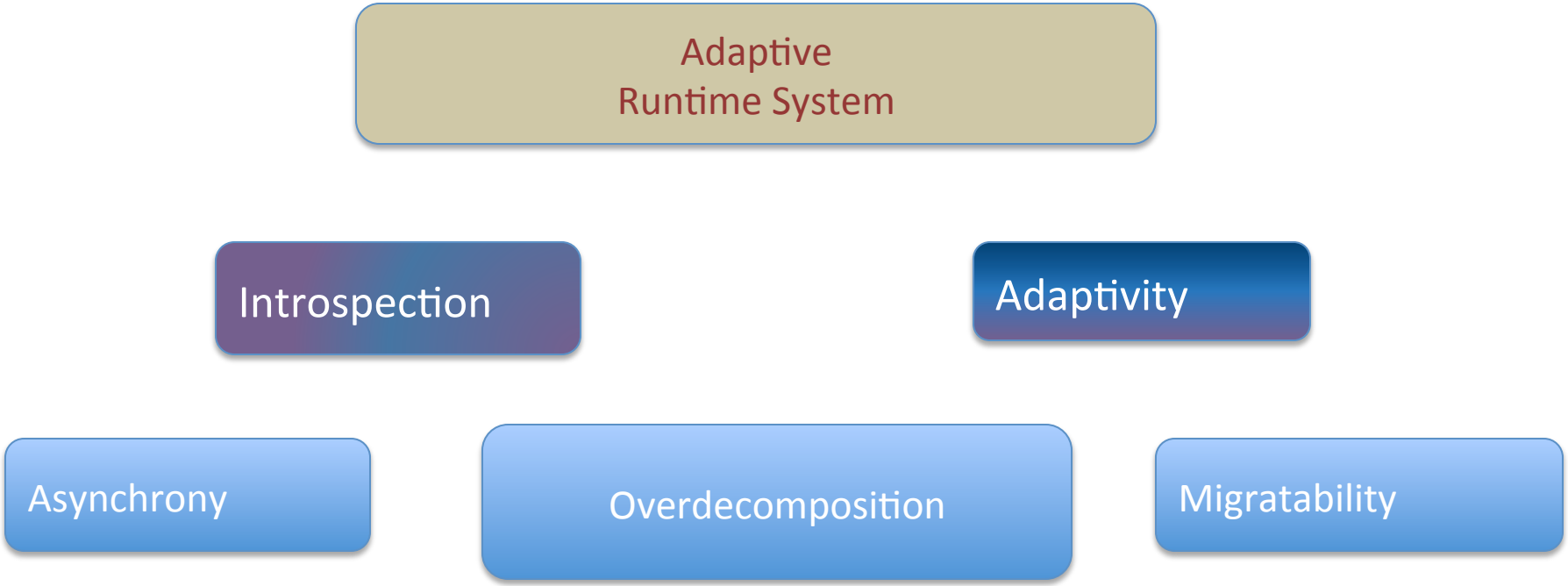



Processor 0
Scheduler
Message Queue
Processor 1
Scheduler
Message Queue
Processor 2
Scheduler
Message Queue
Processor 3
Scheduler
Message Queue


Processor 0
Scheduler
Message Queue
Processor 1
Scheduler
Message Queue
Processor 2
Scheduler
Message Queue
Processor 3
Scheduler
Message Queue


Processor 0
Scheduler
Message Queue
Processor 1
Scheduler
Message Queue
Processor 2
Scheduler
Message Queue
Processor 3
Scheduler
Message Queue

Empowering the RTS



- The Adaptive RTS can:
  - Dynamically balance loads
  - Optimize communication:
    - Spread over time, async collectives
  - Automatic latency tolerance
  - Prefetch data with almost perfect predictability

Benefits in Charm++