A4

SWE-X10: An Actor-Based and Locally Coordinated Solver for the Shallow Water Equations

6th ACM SIGPLAN Workshop on X10

Alexander Pöppl, Prof. Dr. Michael Bader
Technical University of Munich

Motivation

- Wanted: Interesting target For Research in the field of Parallel Algorithms in Scientific Computing:
- Shallow Water Equations
  - Commonly used in the simulation of tsunami events
  - Two-dimensional simulation problem
  - Realistic scenarios may be solved with a moderate amount of resources
  - Not embarrasingly parallel
- Goal: Combine advantages of APGAS and actor-based programming to maximize flexibility
  - Asynchronous local time stepping scheme
  - Support for heterogeneous hardware platforms
  - Simple exploration of different configurations

The Shallow Water Equations

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = S(t, x, y)$$

$h(x, y, t)$	Water Height
$b(x, y)$	Bathymetry (Sea floor & terrain structure)
$u(x, y, t)$	Horizontal fluid velocity (x direction, depth-averaged)
$v(x, y, t)$	Horizontal fluid velocity (y direction, depth-averaged)
$hu(x, y, t)$	Momentum in x direction
$hv(x, y, t)$	Momentum in y direction

The Shallow Water Equations

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = S(t, x, y)$$

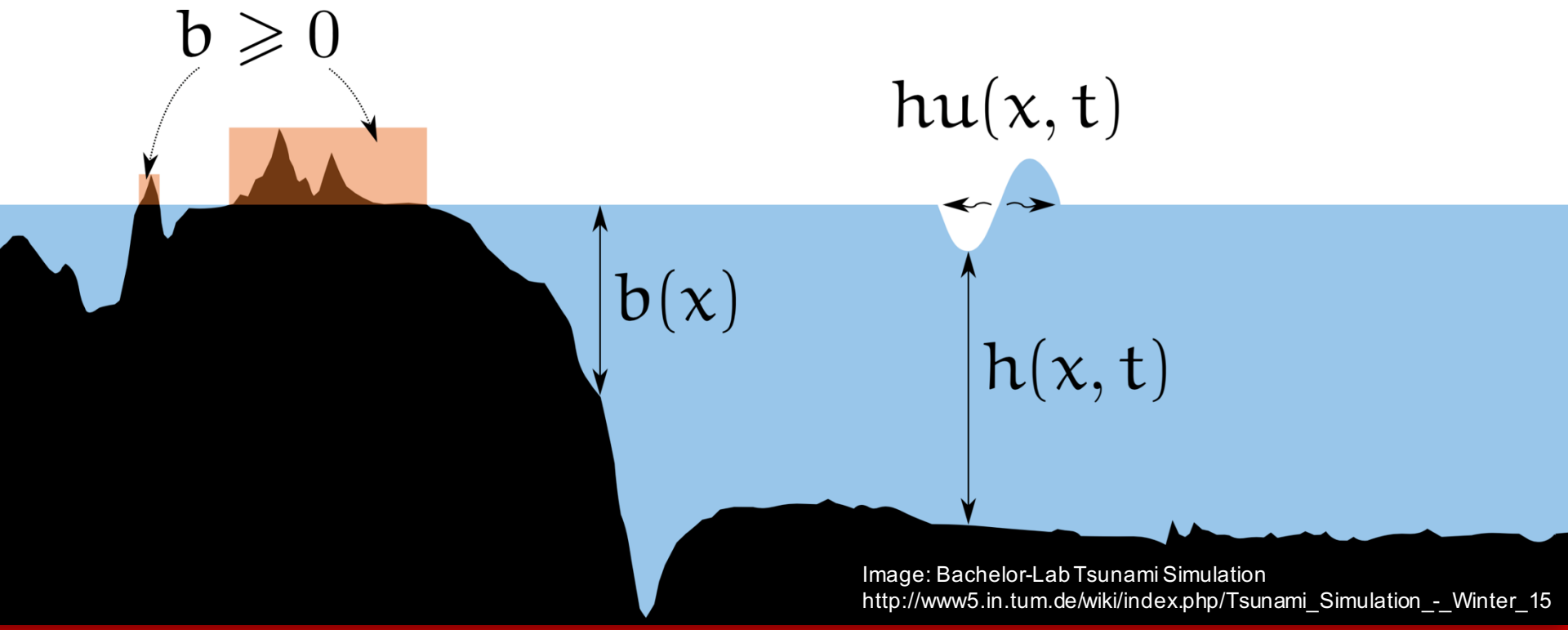
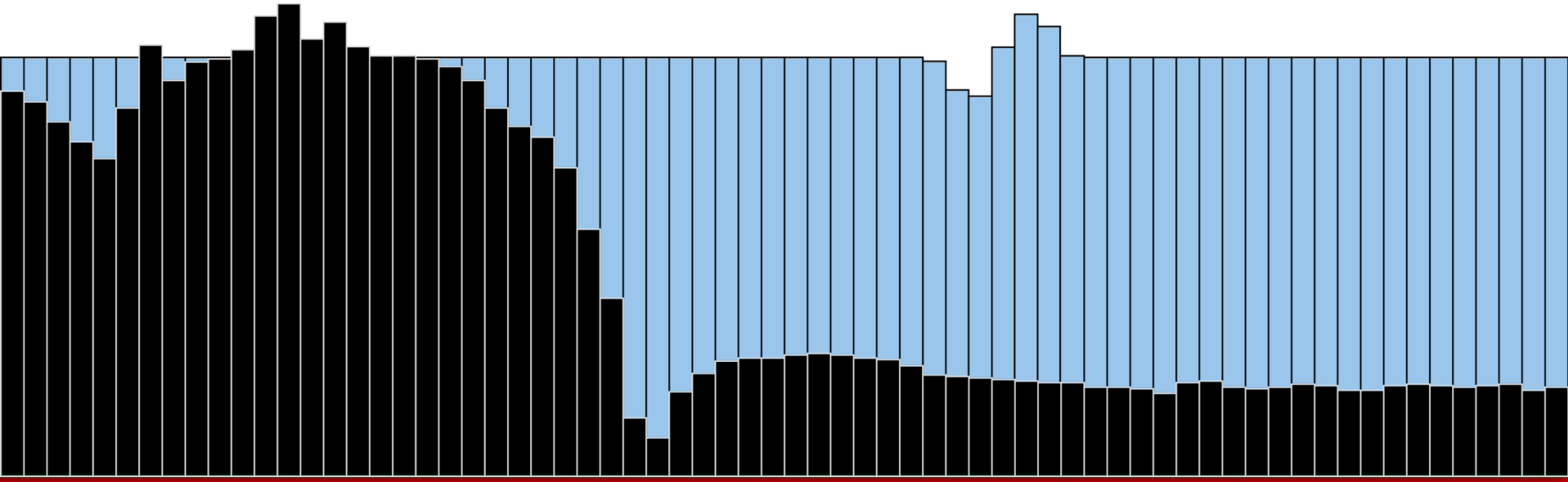


Image: Bachelor-Lab Tsunami Simulation
http://www5.in.tum.de/wiki/index.php/Tsunami_Simulation_-_Winter_15

Shallow Water Equations

$$Q_{i,j}^{n+1} = Q_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^{+}\Delta Q_{i-\frac{1}{2},j} + \mathcal{A}^{-}\Delta Q_{i+\frac{1}{2},j}^{n}\right) - \frac{\Delta t}{\Delta y}\left(\mathcal{B}^{+}\Delta Q_{i,j-\frac{1}{2}} + \mathcal{B}^{-}\Delta Q_{i,j+\frac{1}{2}}^{n}\right)$$



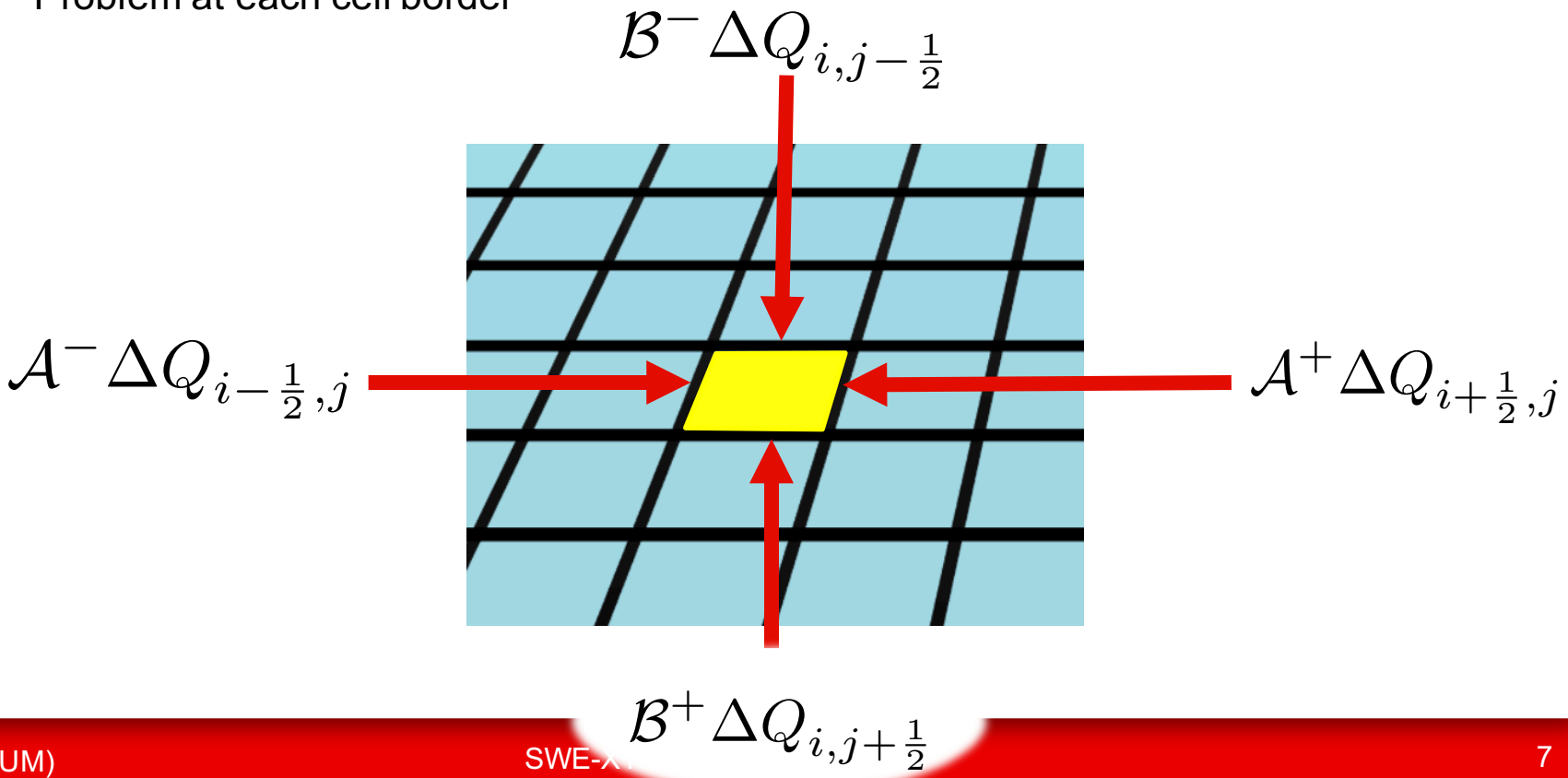
Shallow Water Equations

$$Q_{i,j}^{n+1} = Q_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^{+}\Delta Q_{i-\frac{1}{2},j} + \mathcal{A}^{-}\Delta Q_{i+\frac{1}{2},j}^{n}\right) - \frac{\Delta t}{\Delta y}\left(\mathcal{B}^{+}\Delta Q_{i,j-\frac{1}{2}} + \mathcal{B}^{-}\Delta Q_{i,j+\frac{1}{2}}^{n}\right)$$

$Q_{i,j}^{n} = [h_{i,j}, hu_{i,j}, hv_{i,j}]$	Observed Quantities at time step t_n
Δt	Length of a timestep
$\Delta x, \Delta y$	Cell size in x,y direction
$\mathcal{A}^{\pm}\Delta Q_{i\pm\frac{1}{2},j}$	Solution of Riemann problem at left/right edge of cell
$\mathcal{B}^{\pm}\Delta Q_{i,j\pm\frac{1}{2}}$	Solution of Riemann problem at top/bottom edge of cell

SWE-X10

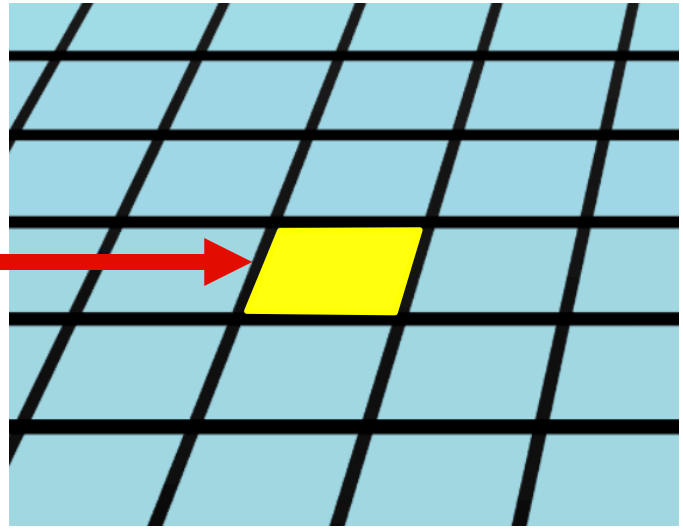
- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border



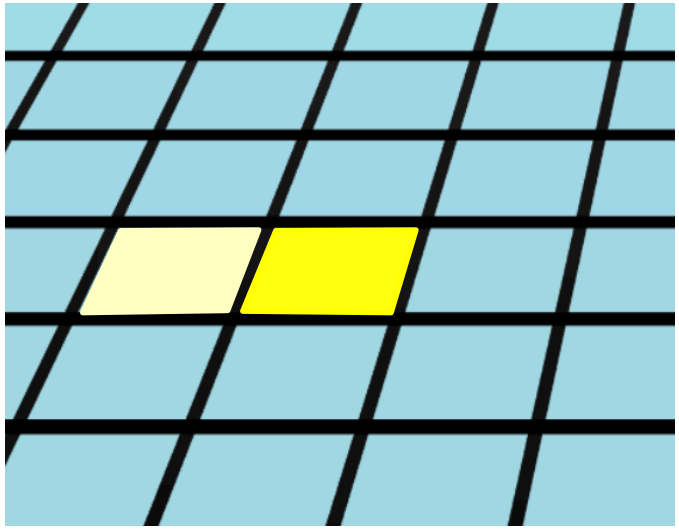
SWE-X10

- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border

$$\mathcal{A}^{-}\Delta Q_{i-\frac{1}{2},j}$$

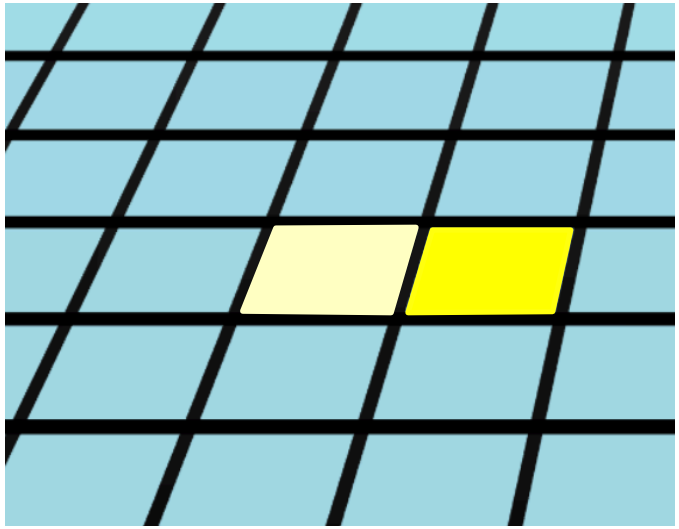


- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border



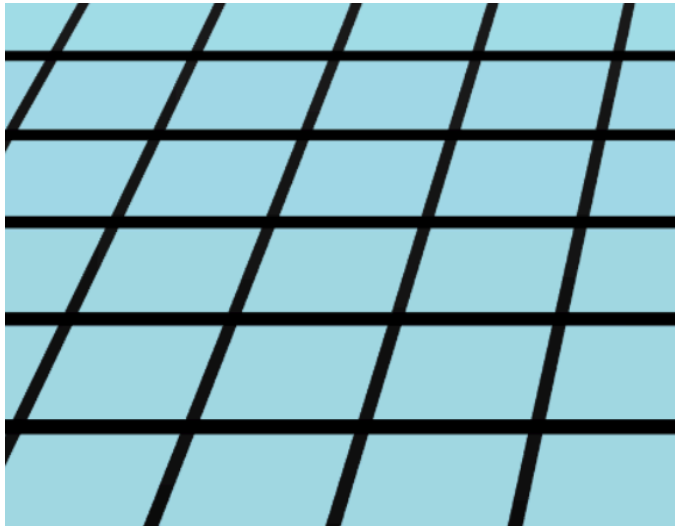
SWE-X10

- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border



SWE-X10

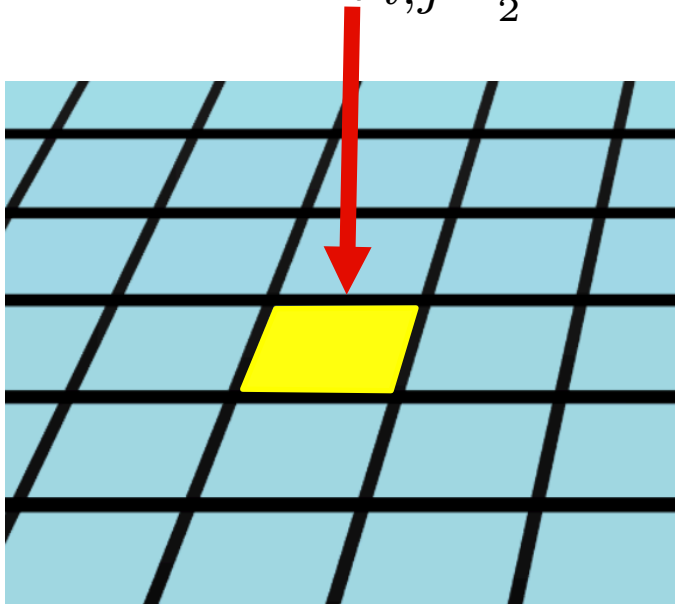
- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border



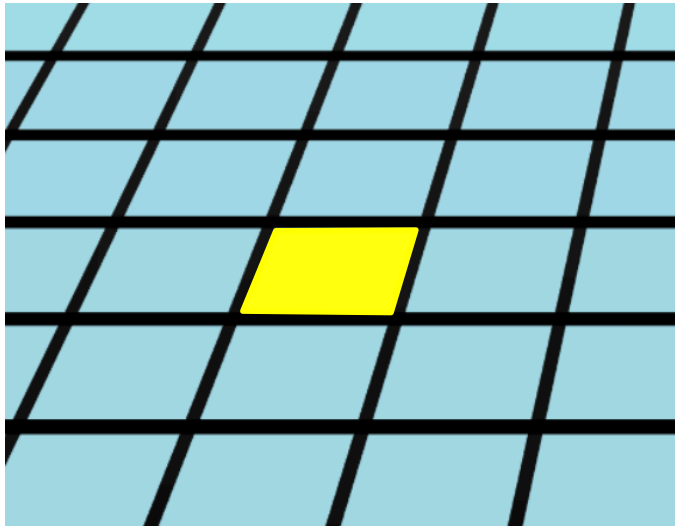
SWE-X10

- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border

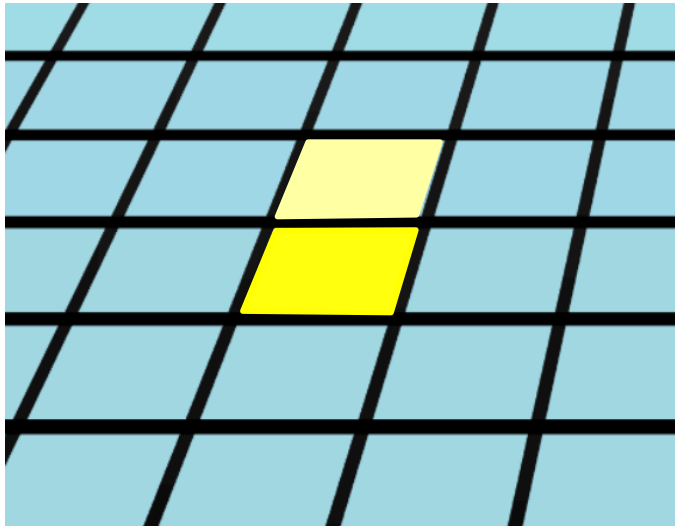
$$\mathcal{B}^{-}\Delta Q_{i,j-\frac{1}{2}}$$



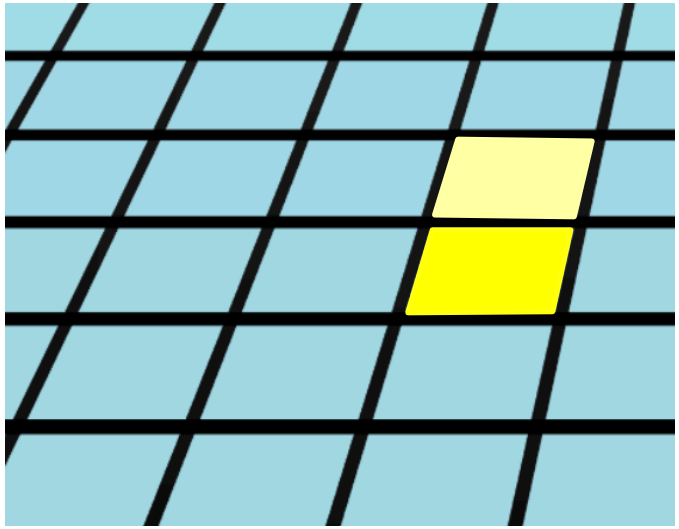
- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border



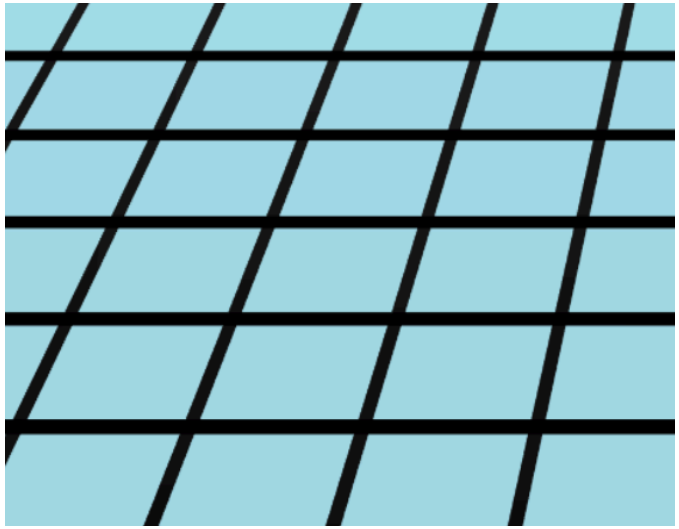
- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border



- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border

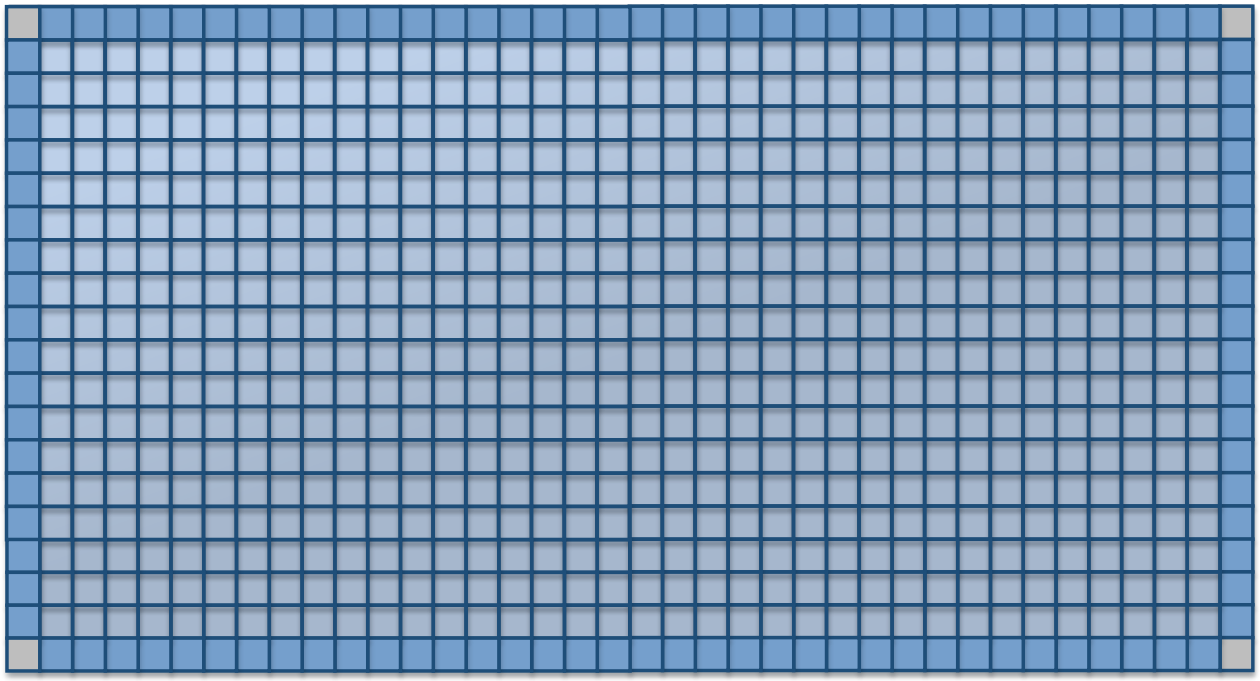


- Two-dimensional Grid
  - Uniformly sized grid cells
  - Update scheme known
  - Iteration over whole grid, compute updates for each grid cell by solving the Riemann Problem at each cell border



Mapping onto Actor Model

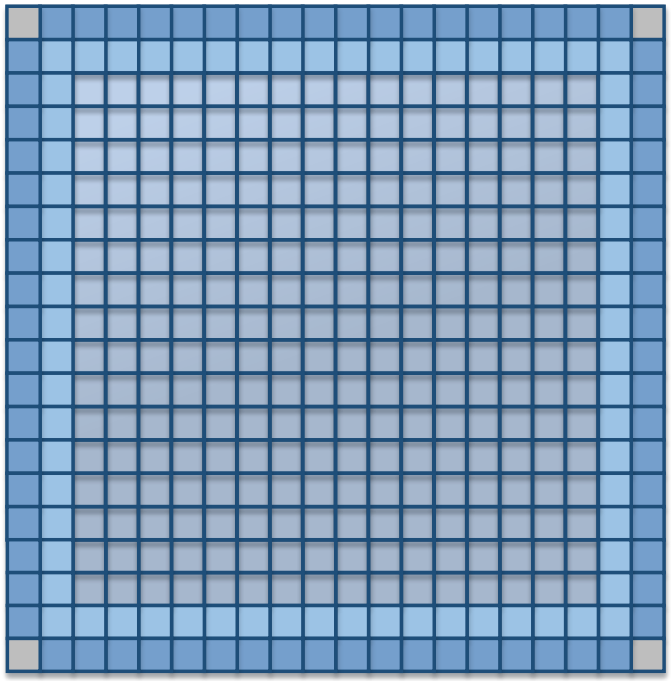
A4

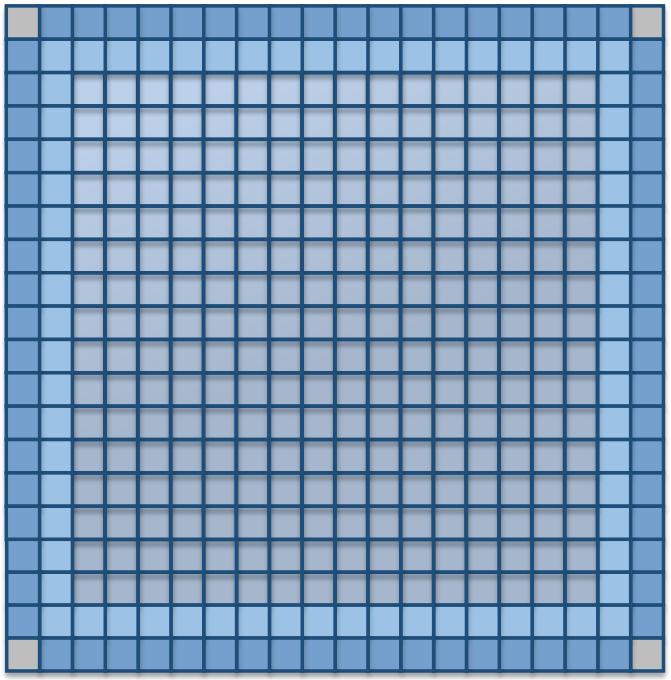


Mapping onto Actor Model

A4

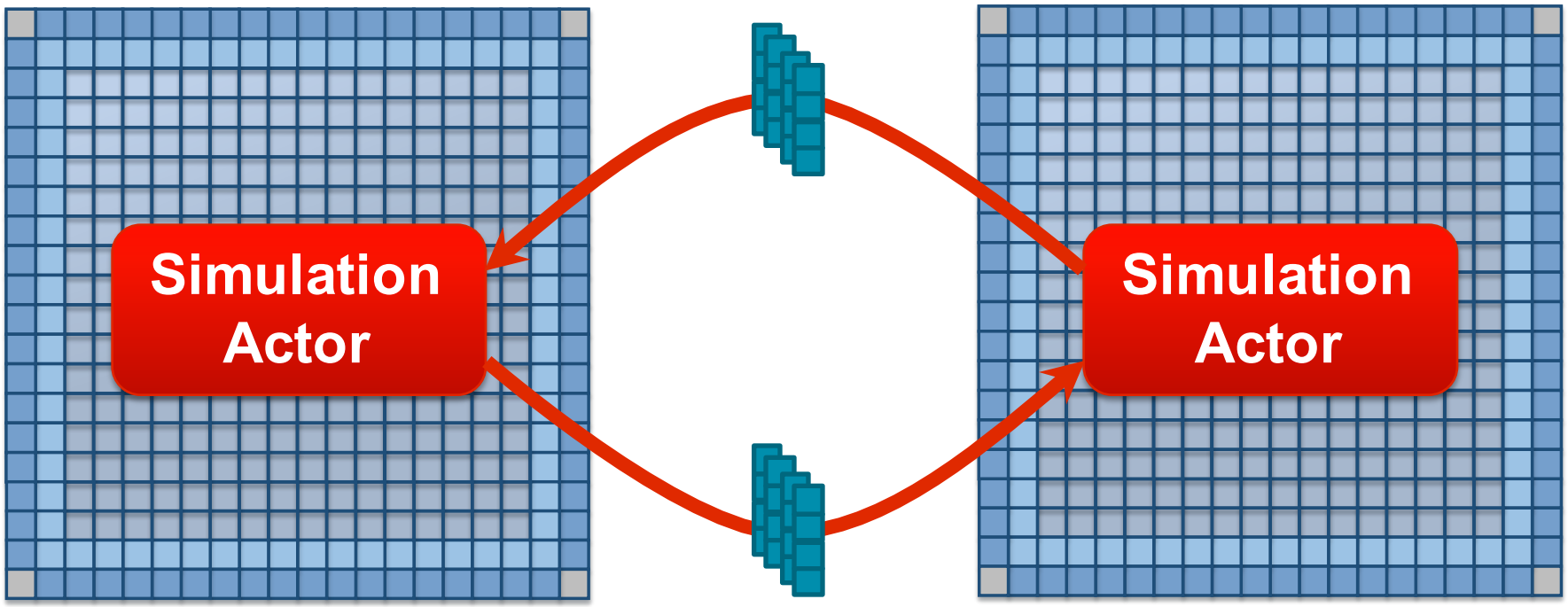
- Subdivision of simulation domain into rectangular patches
- Each patch replicates boundary layers of neighboring Patches
  - Data Exchange after each time step
- Coordination using actors





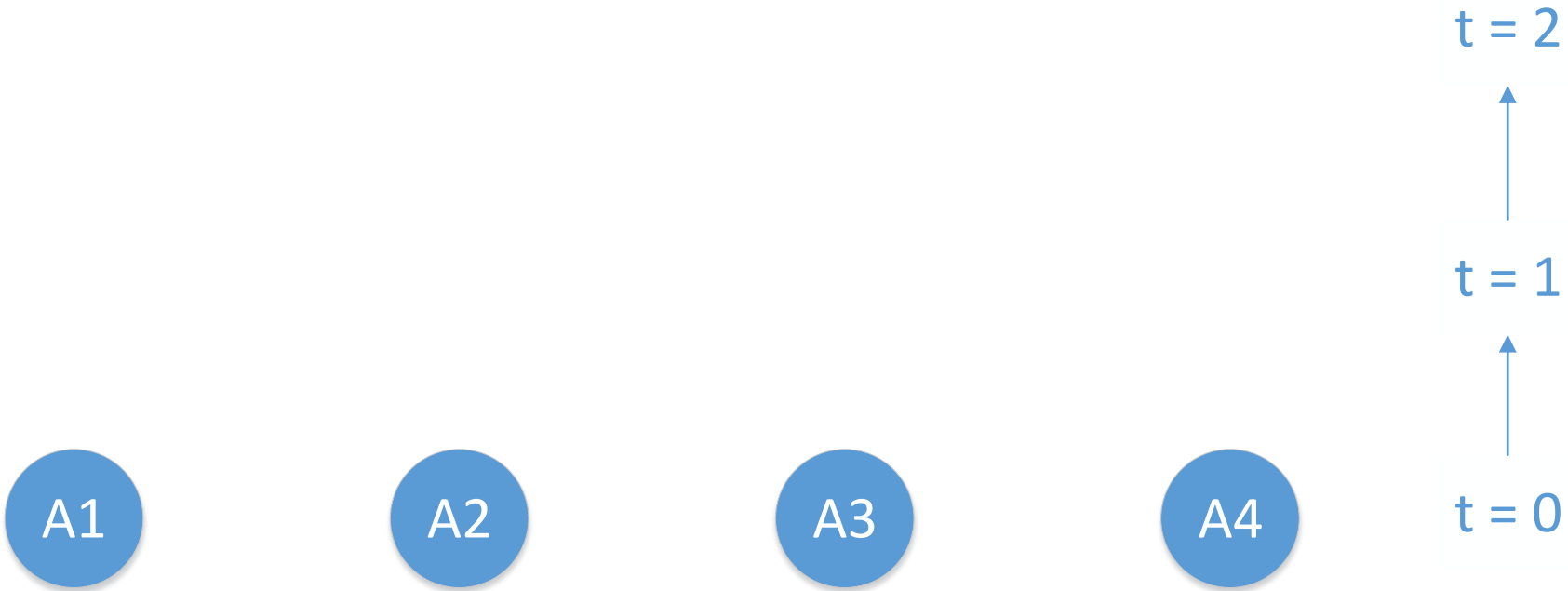
Mapping onto Actor Model

- Subdivision of simulation domain into rectangular patches
- Each patch replicates boundary layers of neighboring Patches
  - Data Exchange after each time step
- Coordination using actors



Actor Model – Local Coordination

A4

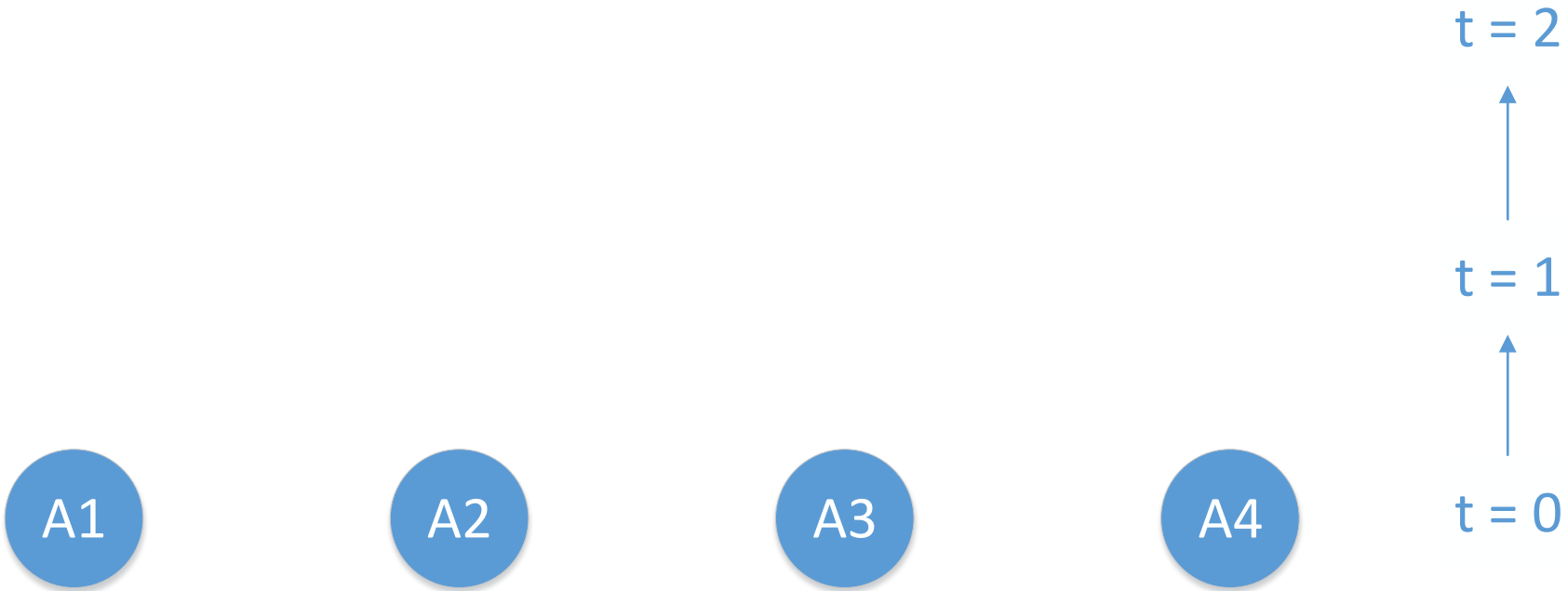


Actor Model – Local Coordination

A4

Actor Model – Local Coordination

A4

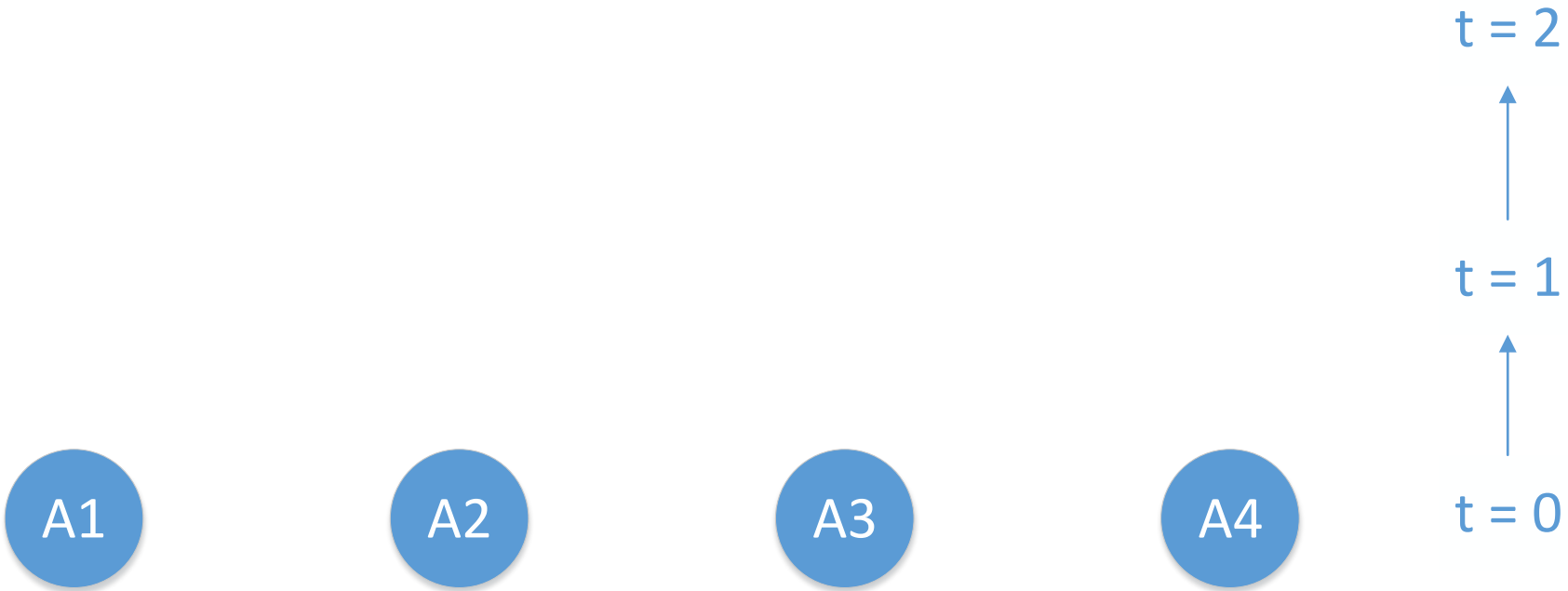


Actor Model – Local Coordination

A4

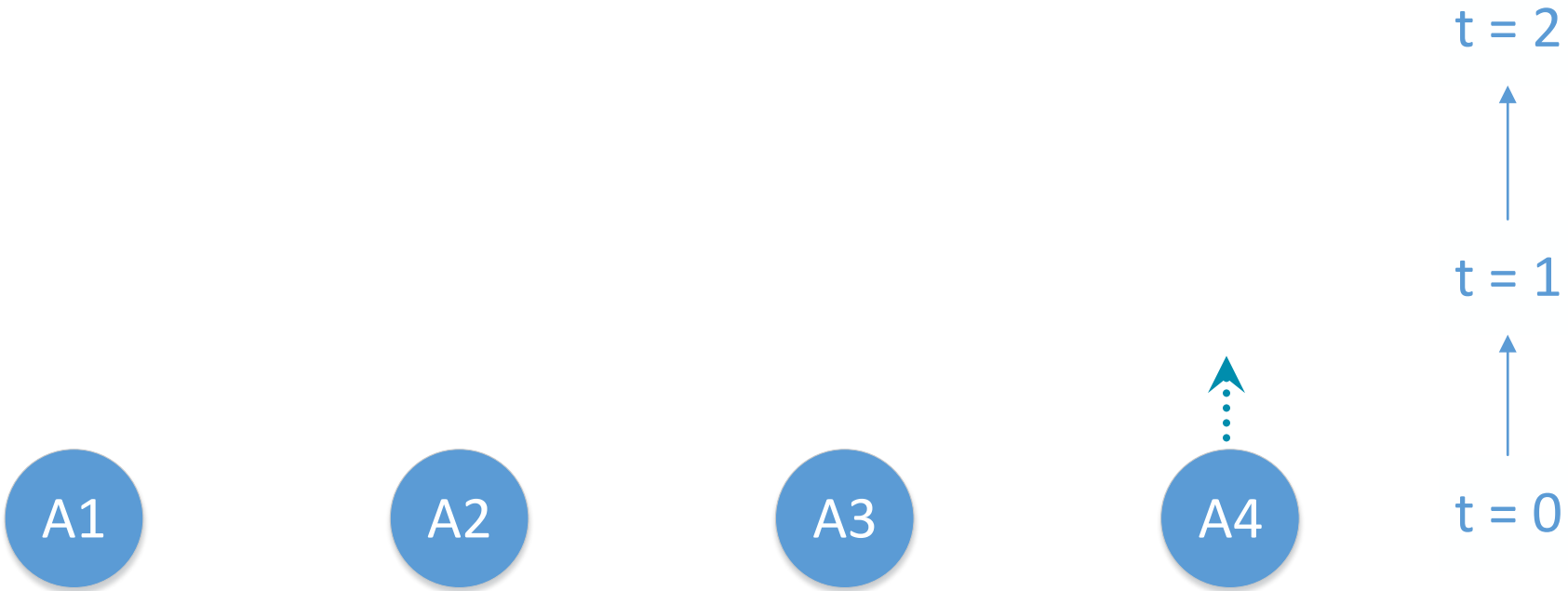
Actor Model – Local Coordination

A4

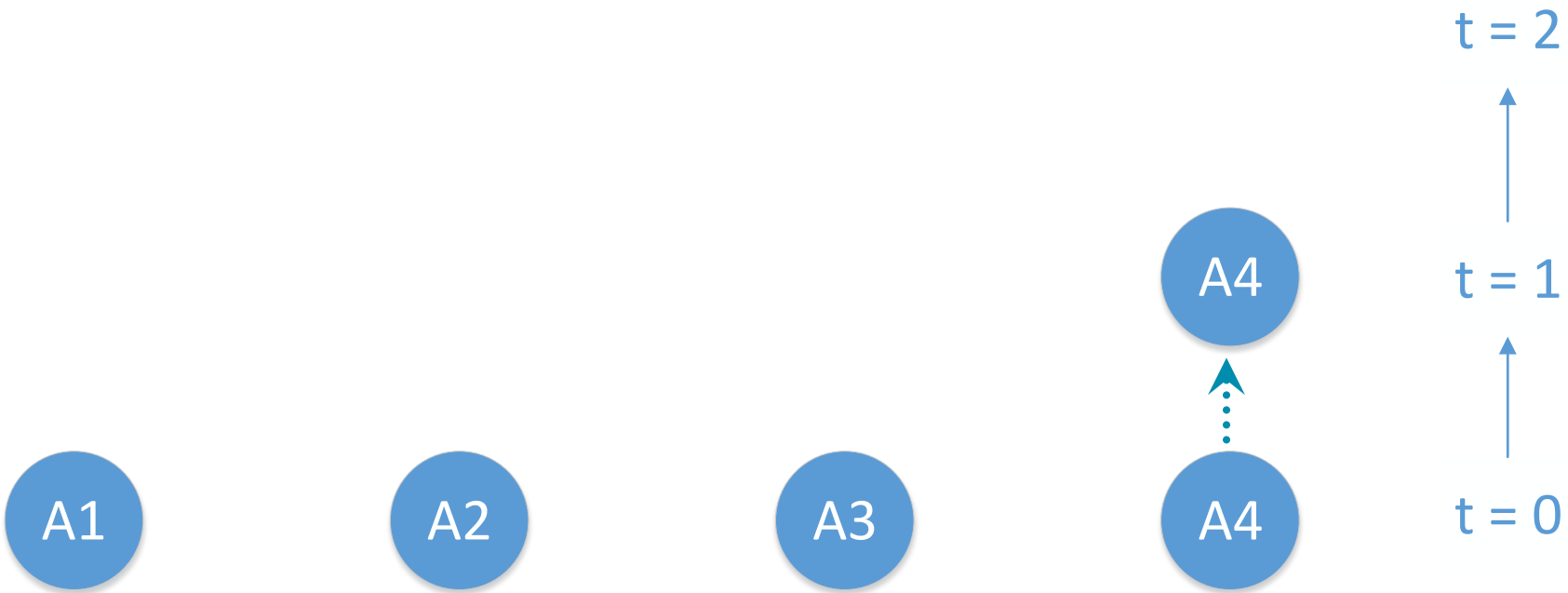


Actor Model – Local Coordination

A4



Actor Model – Local Coordination

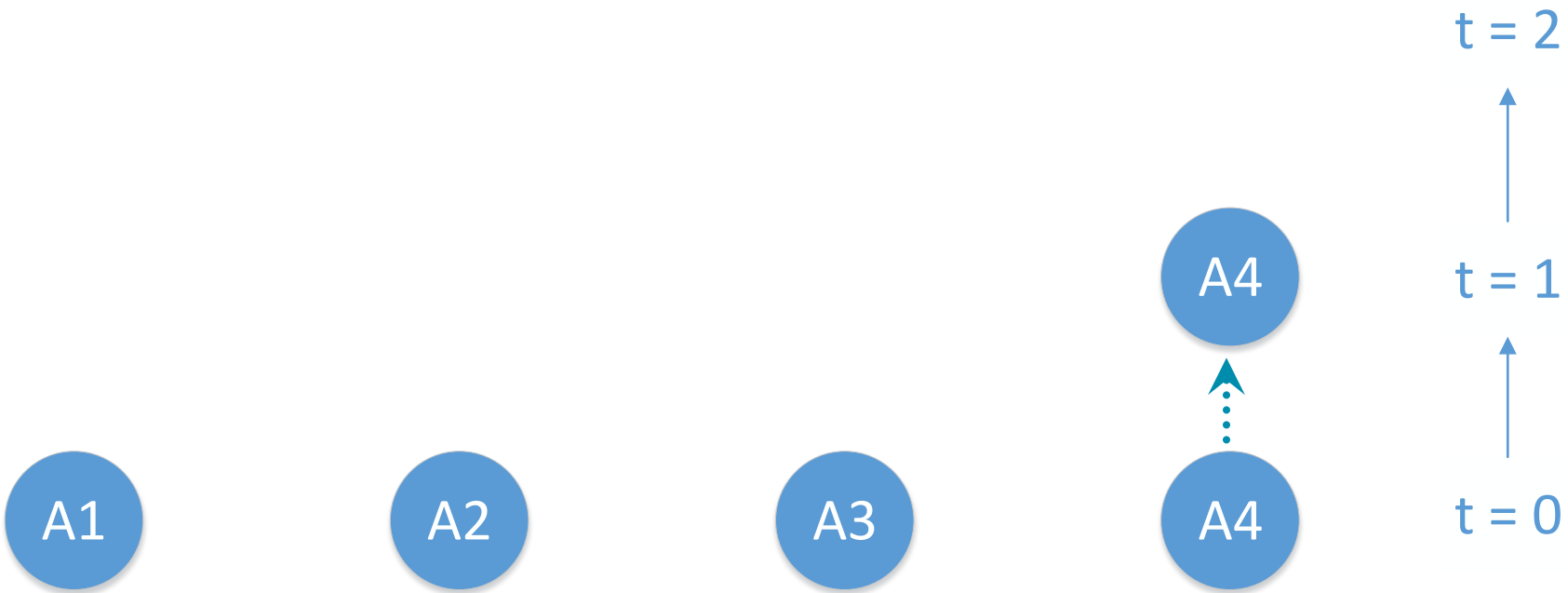


Actor Model – Local Coordination

A4

Actor Model – Local Coordination

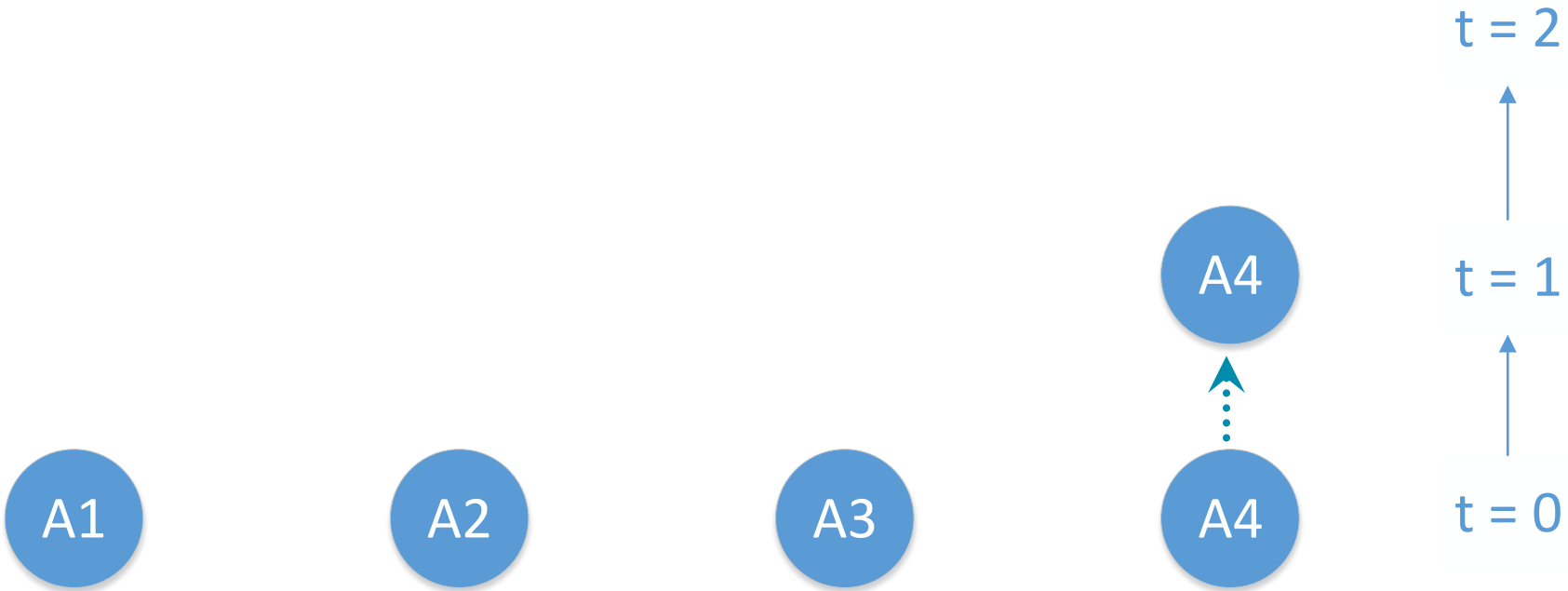
A4



Actor Model – Local Coordination

A4

Actor Model – Local Coordination

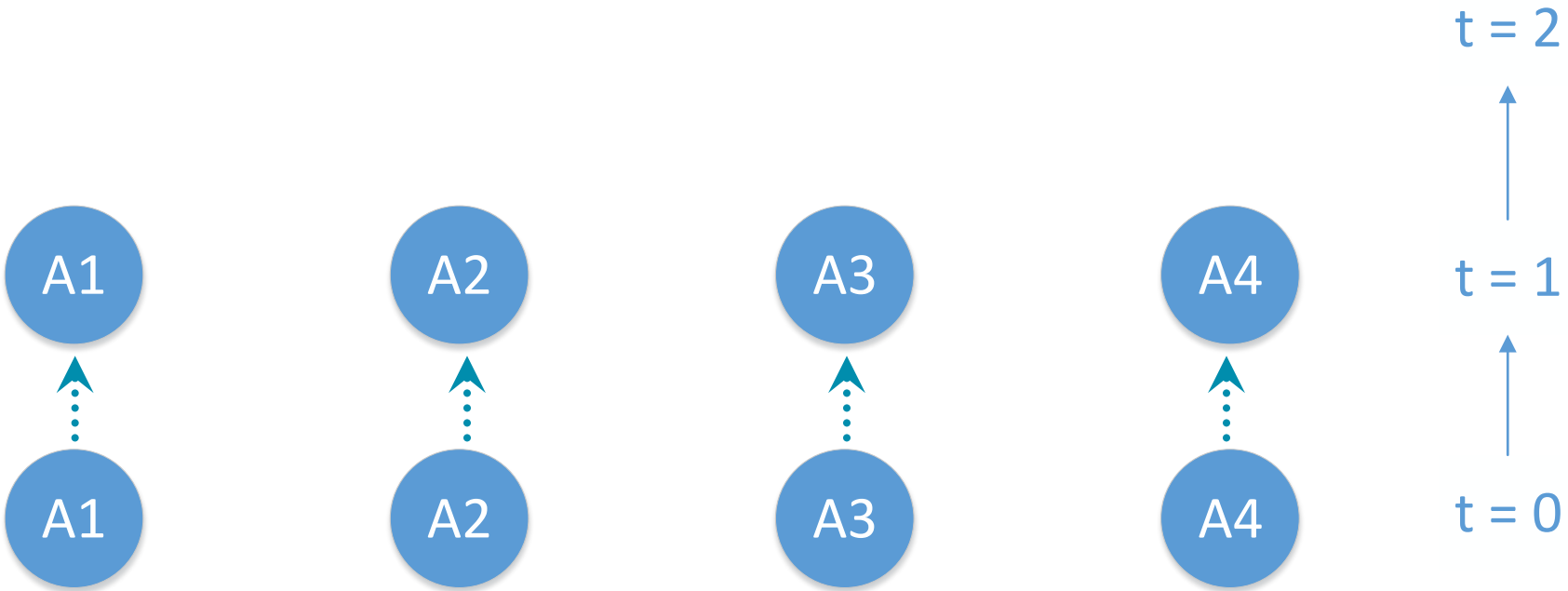


Actor Model – Local Coordination

A4

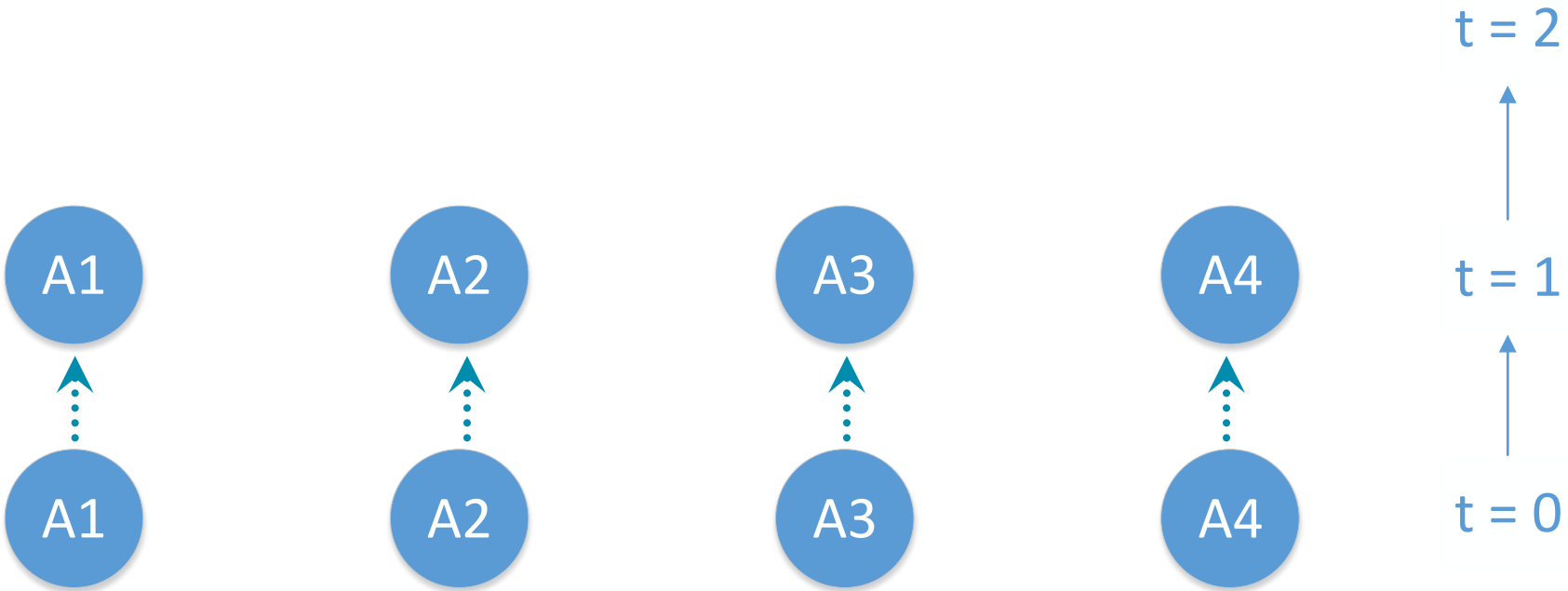
Actor Model – Local Coordination

A4



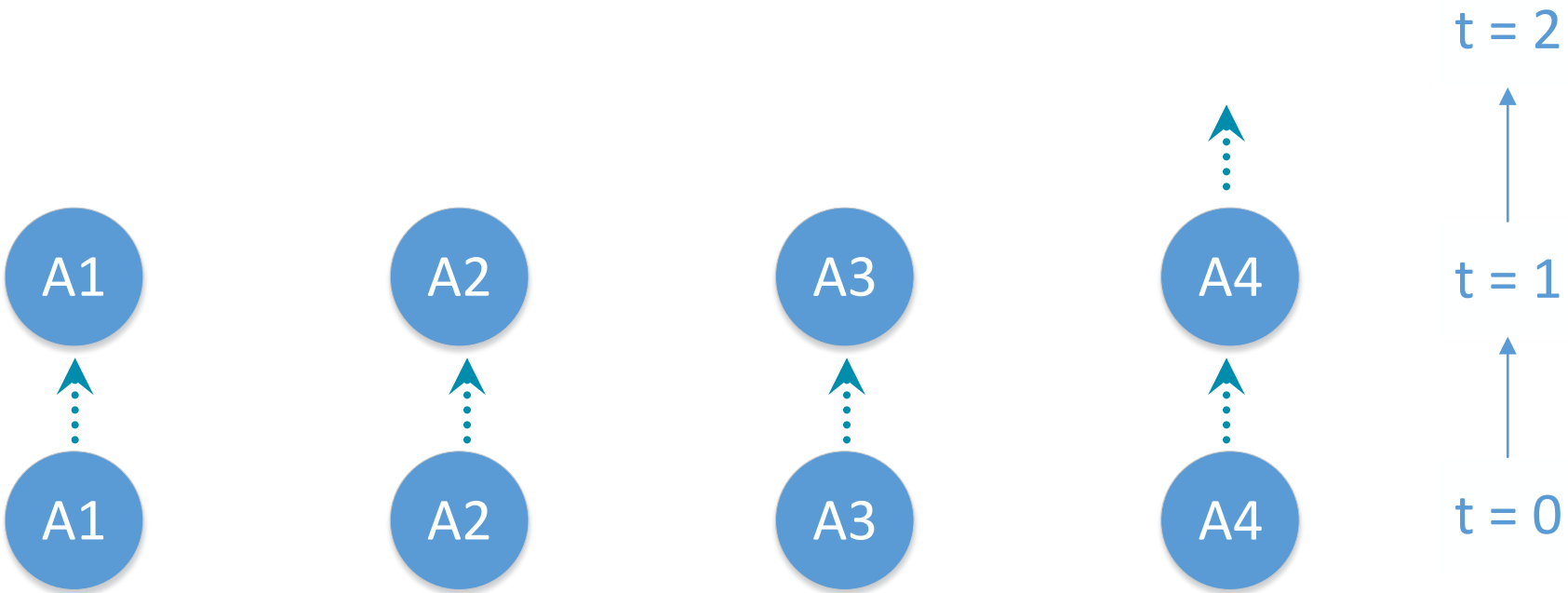
Actor Model – Local Coordination

A4



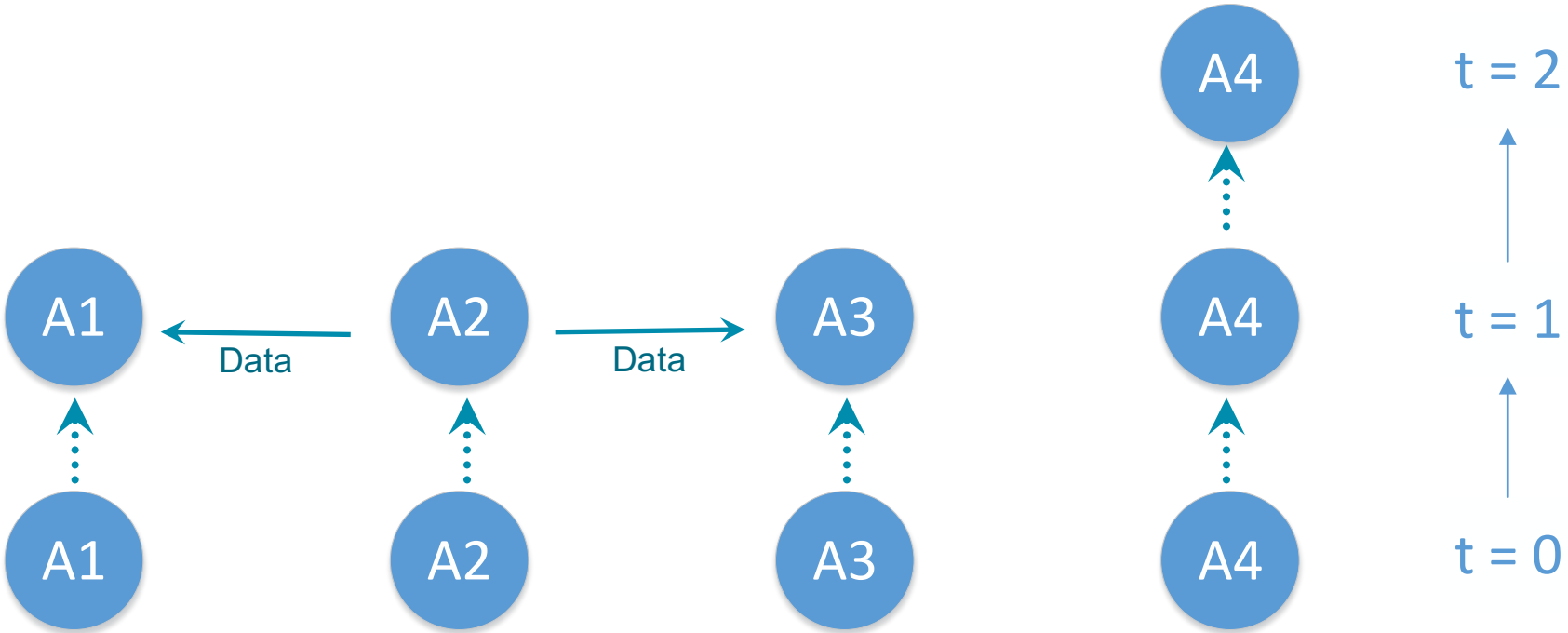
Actor Model – Local Coordination

A4



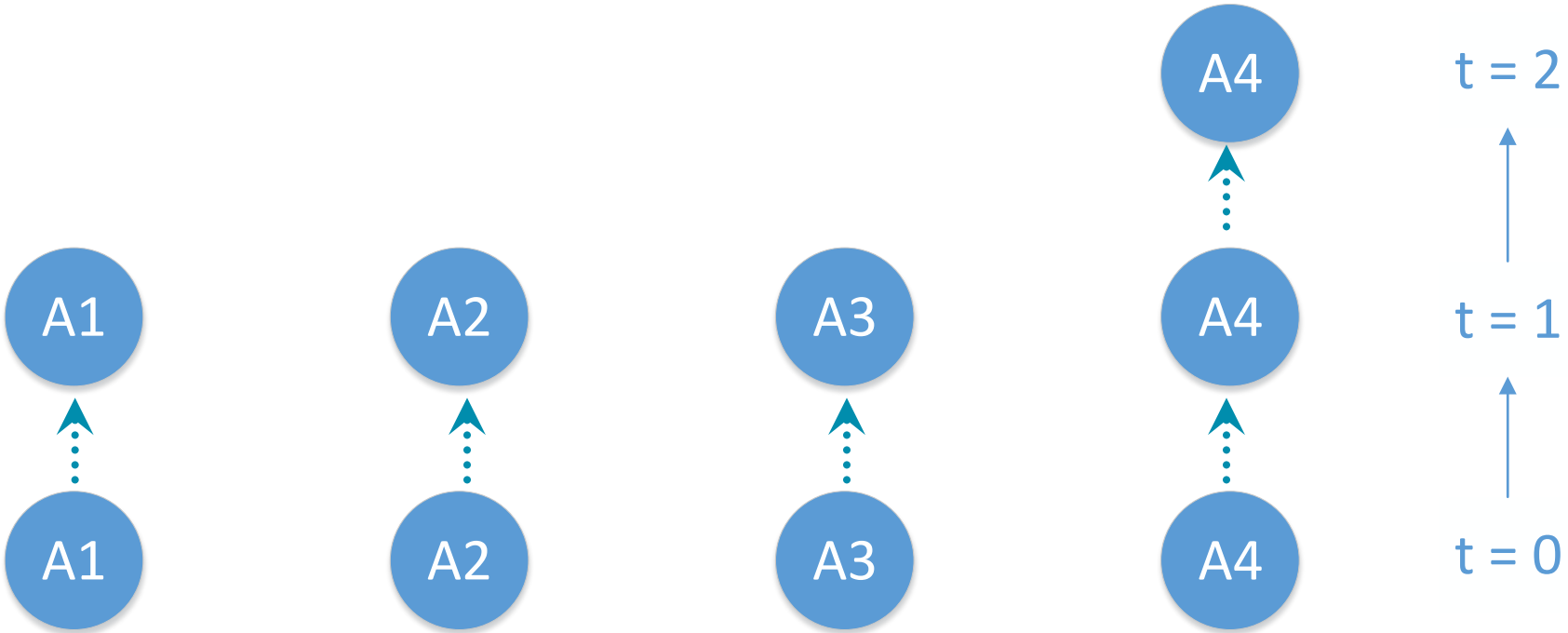
Actor Model – Local Coordination

A4



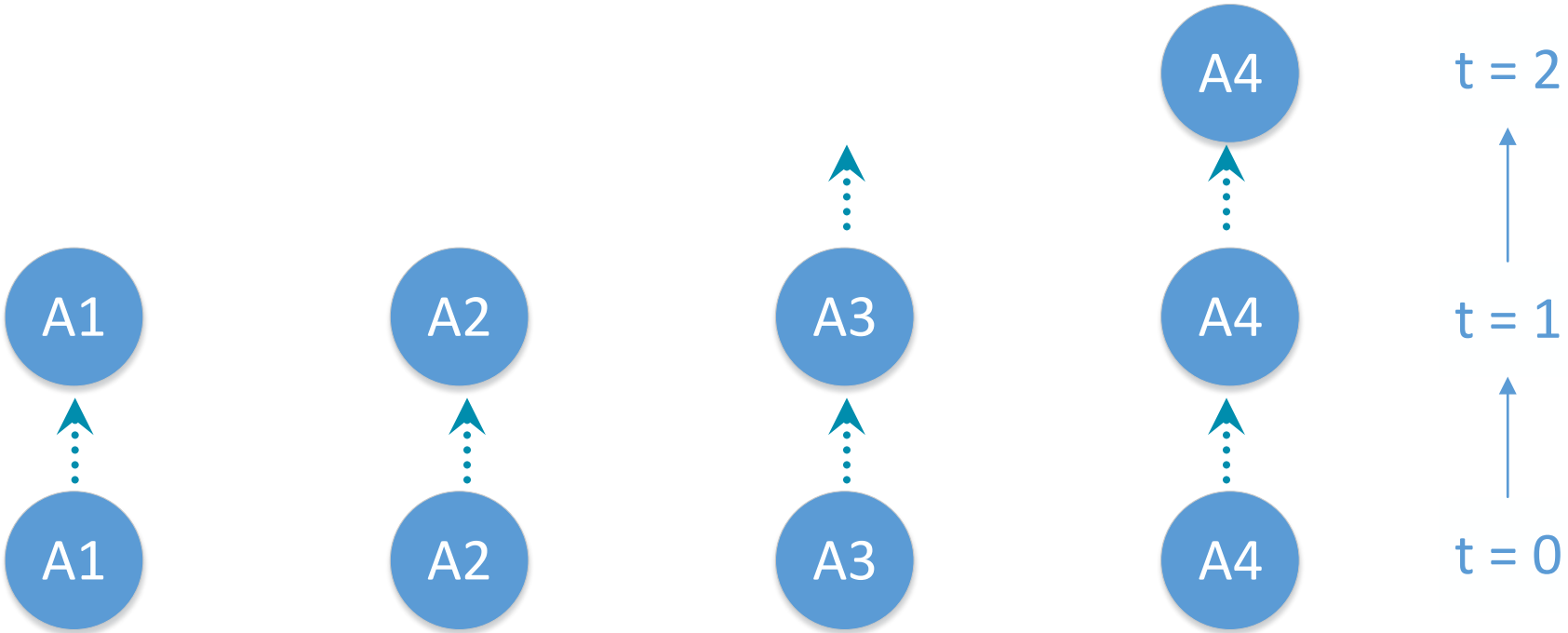
Actor Model – Local Coordination

A4

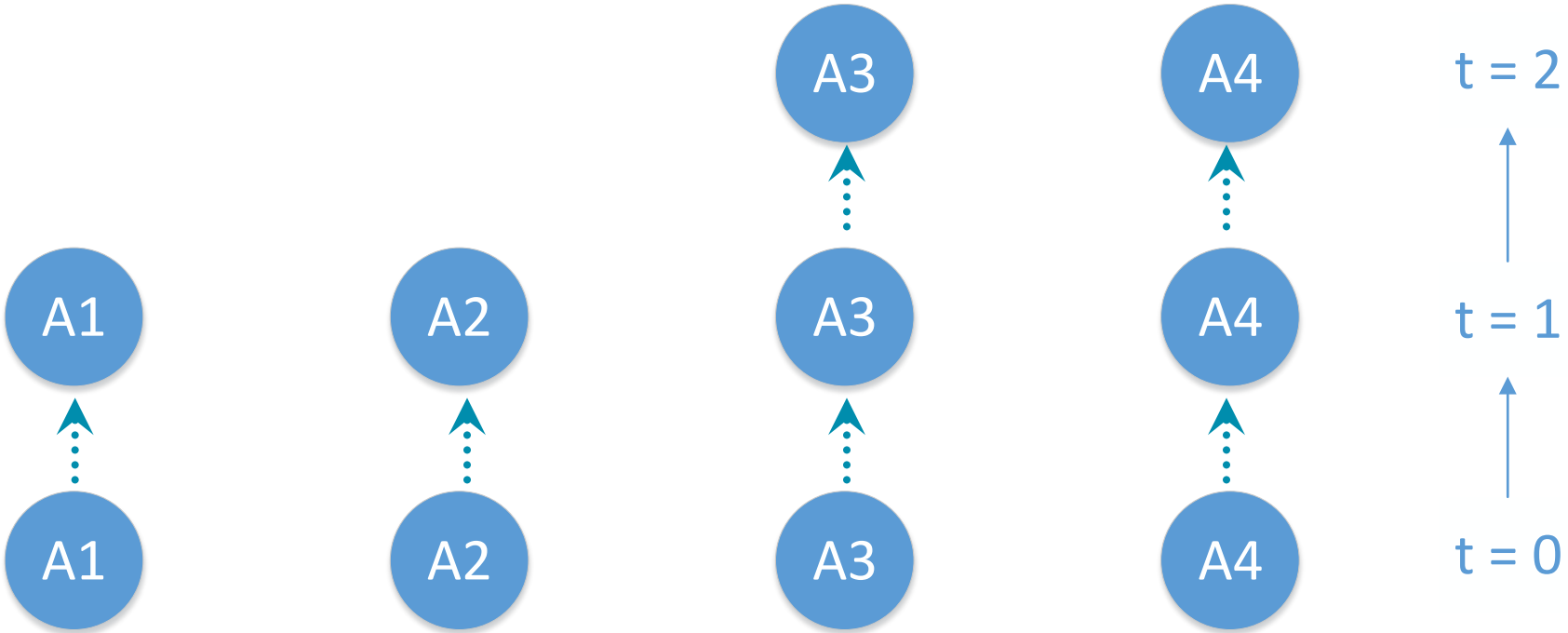


Actor Model – Local Coordination

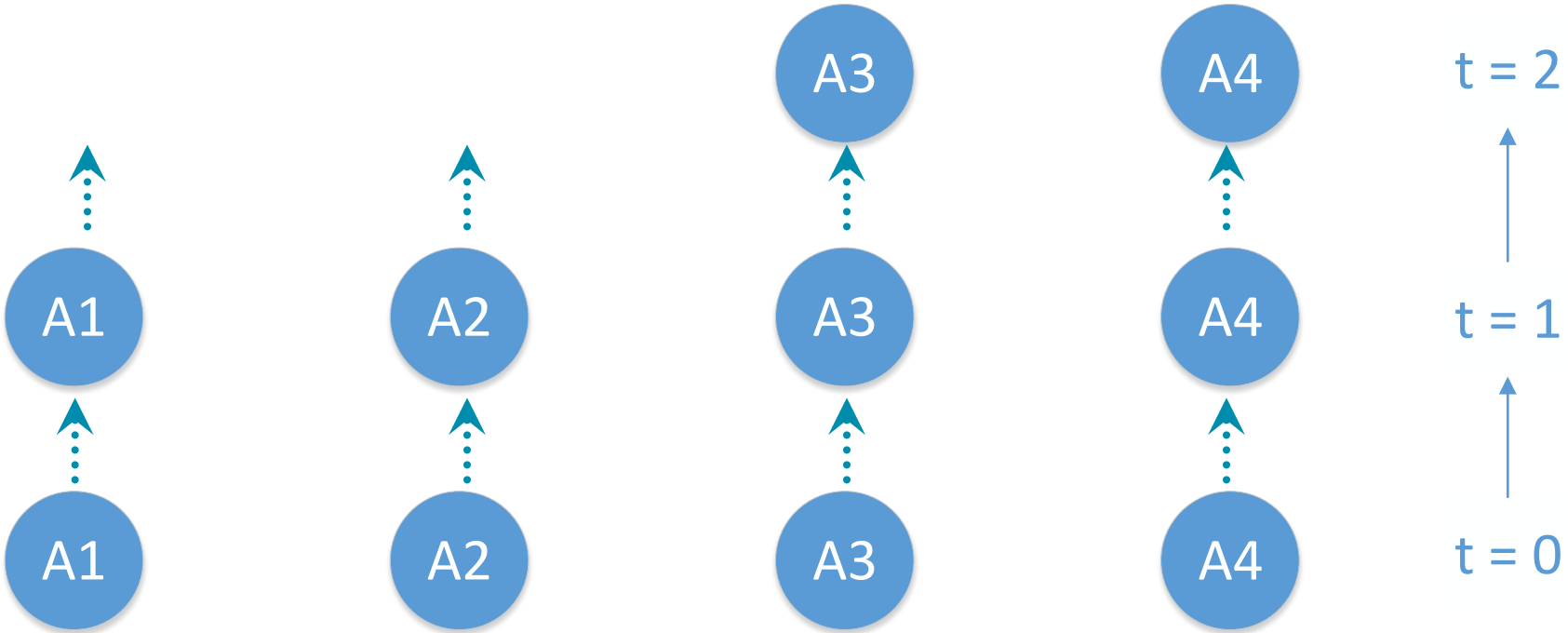
A4



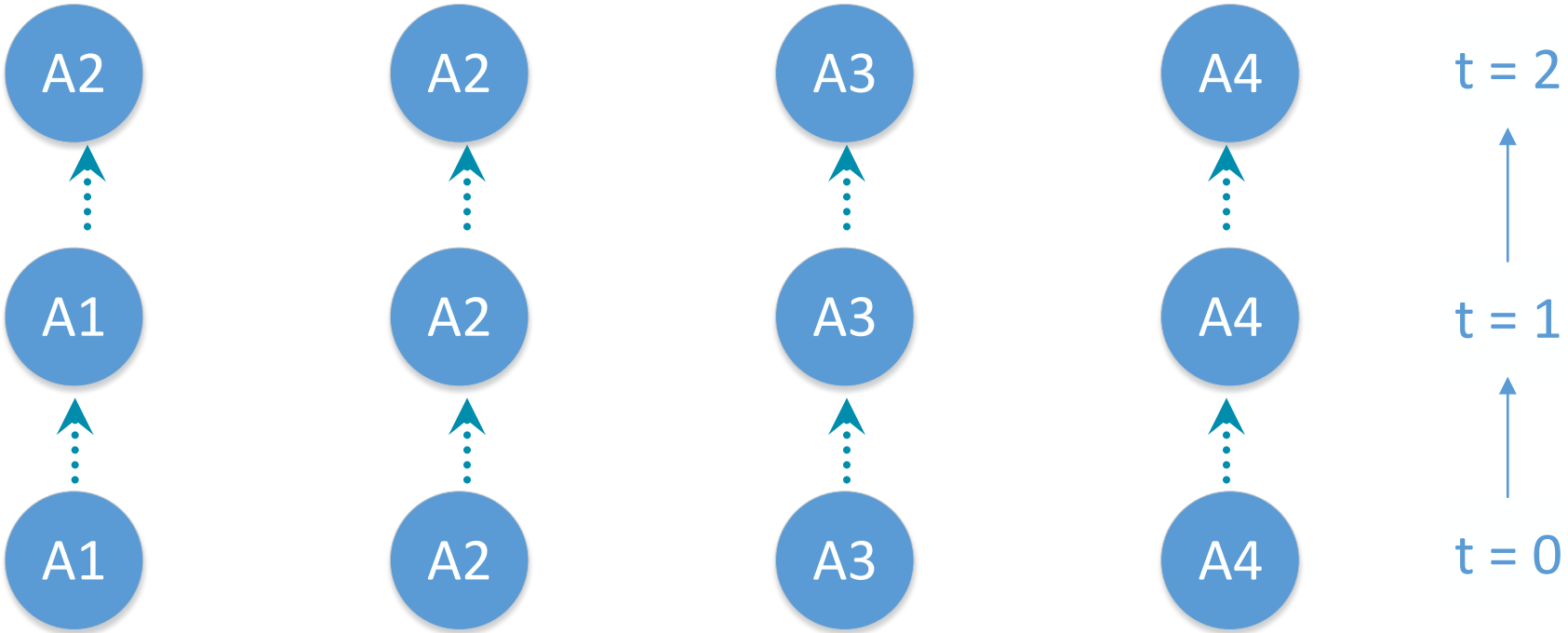
Actor Model – Local Coordination



Actor Model – Local Coordination



Actor Model – Local Coordination



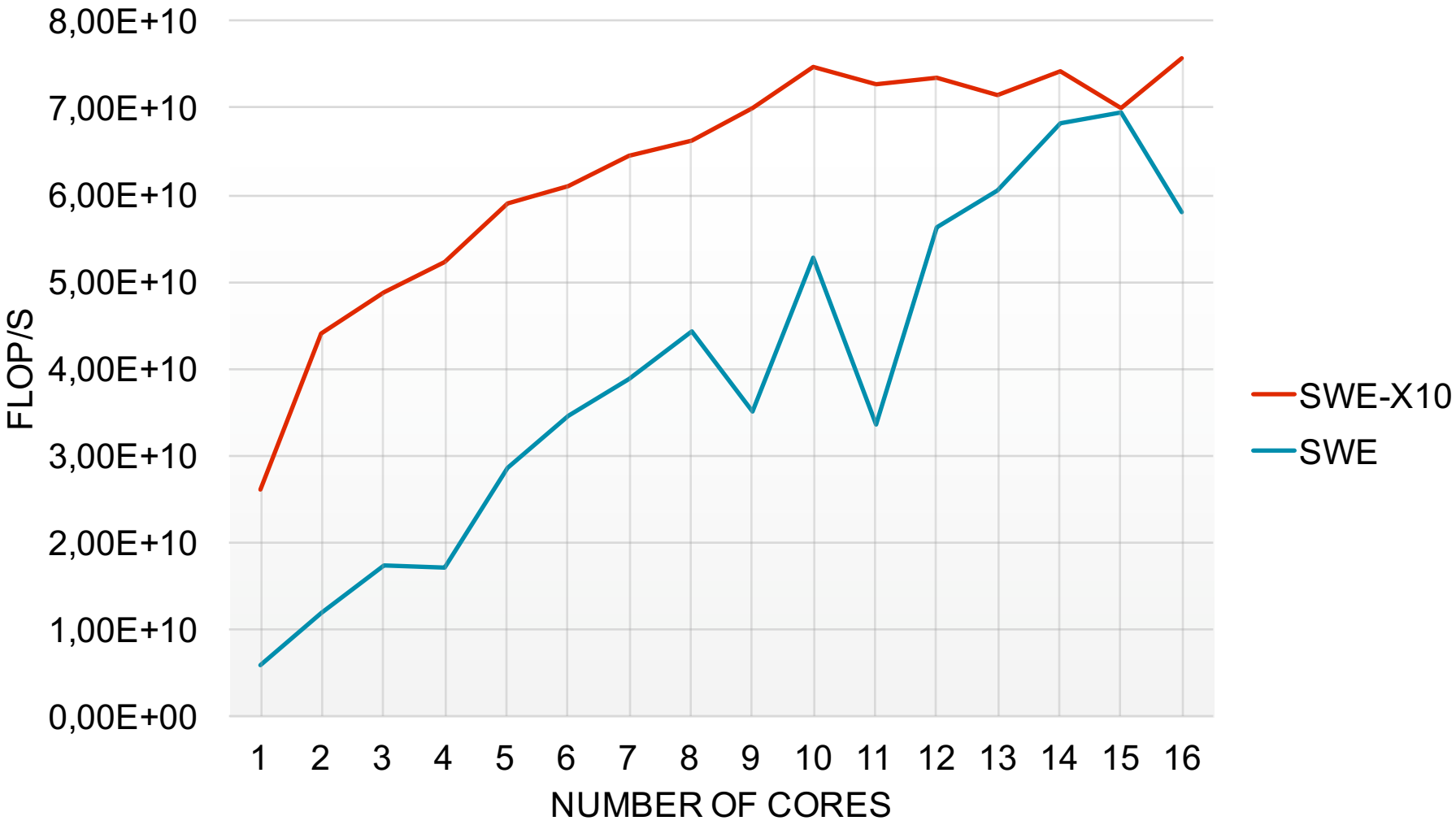
Single Node – Test Setup

A4

- Single node in a cluster
  - Dual Socket Sandy Bridge EP (Xeon E5-2670)
  - 128GB memory
  - Node Peak Single Precision Floating Point Performance: 332.8GFlop/s
  - Memory Bandwidth (STREAM Triad): 60.8GB/s
  - Intel C++ Compiler 16.0
- 1024x1024 cells per CPU core
  - 4 actors with 512x512 cells each
- SWE[1] (OpenMP+MPI-based Solver) as comparison
  - https://github.com/TUM-I5/SWE

1. A. Breuer and M. Bader. Teaching parallel programming models on a shallow-water code. In *Proceedings of the 2012 11th International Symposium on Parallel and Distributed Computing*, ISPDC '12, pages 301–308, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4805-0. doi: 10.1109/ISPDC.2012.48. URL http://dx. doi.org/10.1109/ISPDC.2012.48.

Single Node – Test Results



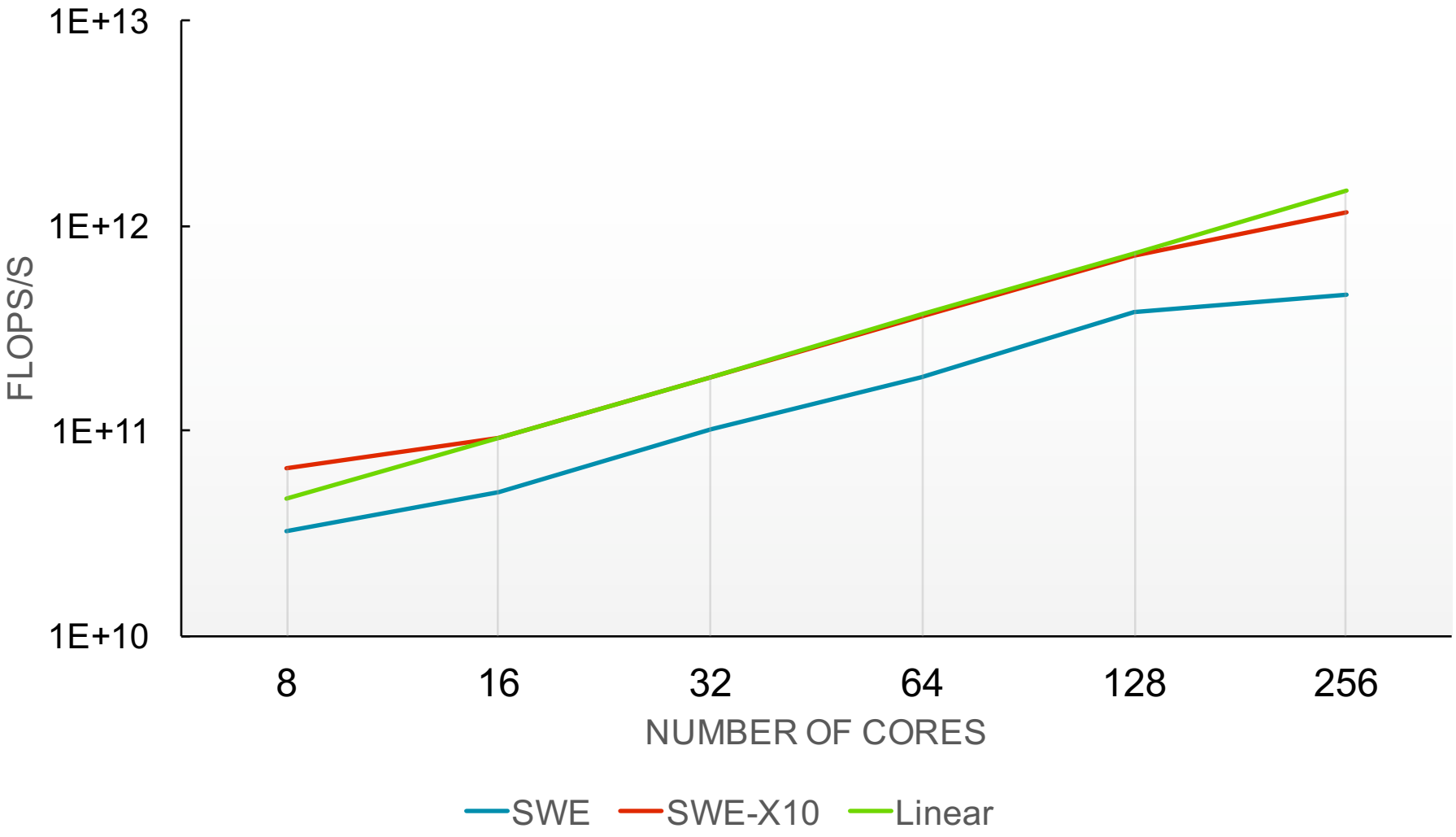
Multiple Nodes – Test Setup

- Cluster with 28 nodes
  - Dual Socket Sandy Bridge EP (Xeon E5-2670)
  - 128GB memory
  - Node Peak Single Precision Floating Point Performance: 332.8GFlop/s
  - Memory Bandwidth (STREAM Triad): 60.8GB/s
  - Intel C++ Compiler 16.0
  - QDR Infiniband
- One Place per socket
  - 32 actors (with 512x512 cells) per Place
  - Tests ranging from 8 cores (1 Socket) to 256 cores (16 nodes)
- SWE[1] (OpenMP+MPI-based Solver) as comparison
  - https://github.com/TUM-I5/SWE

1. A. Breuer and M. Bader. Teaching parallel programming models on a shallow-water code. In *Proceedings of the 2012 11th International Symposium on Parallel and Distributed Computing*, ISPDC '12, pages 301–308, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4805-0. doi: 10.1109/ISPDC.2012.48. URL http://dx. doi.org/10.1109/ISPDC.2012.48.

Multiple Nodes – Test Results

A4



Conclusions and Future Work

A4

- Demonstrated feasibility of actor-based approach for simulation runs with a moderate number of CPU cores.
- Realistic Proxy application
  - 1.2TFlop/s in run with 256 cores
  - Performance comparable to MPI+OpenMP solution
- Goal: Exploit Actor-based Coordination
  - Local Time Stepping without centralized control
  - Delay activation, only compute non-trivial solutions
  - Heterogeneous actors

Thank you for your attention

Acknowledgments

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89).