SWE-X10: Simulating shallow water waves with lazy activation of patches using ActorX10

Alexander Pöppl[1], Michael Bader[1], Tobias Schwarzer[2], Michael Glaß[2]

[1]Technical University of Munich,

[2]Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

Second International Workshop on Extreme Scale Programming Models and Middleware

Salt Lake City, Nov 18th 2016

Motivation

- Challenge for scientific computing: Optimal utilization of heterogeneous and dynamically changing resources by dynamically changing work loads.
- Today: Many scientific applications are based on programming models targeting static and homogeneous resources (SPMD pattern)
- Different programming paradigms may open new perspectives to look at problems.

Our approach:

- APGAS programming paradigm (X10 programming language)
- Actor-based modelling (ActorX10 framework)
- Low-level optimization of compute kernel (Auto-Vectorizing C++ loops)

Motivation

Proxy-Application SWE-X10

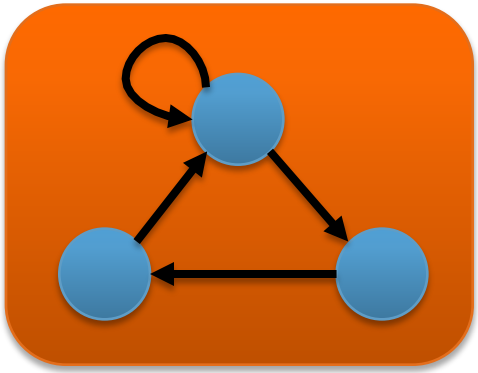
- Built on ActorX10
- Control based on finite state machines
- No global coordination scheme, actors schedule themselves
- Lazy activation of patches

Goal

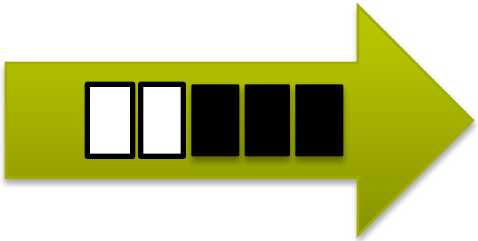
- Show flexibility of actor-based approach
- Provide base for further extensions

Actors and Channels

Actors
- “Threads + Objects”
- Semantics of Execution defined in terms of Functions and Finite State Machines
- State transitions triggered by changes in channels



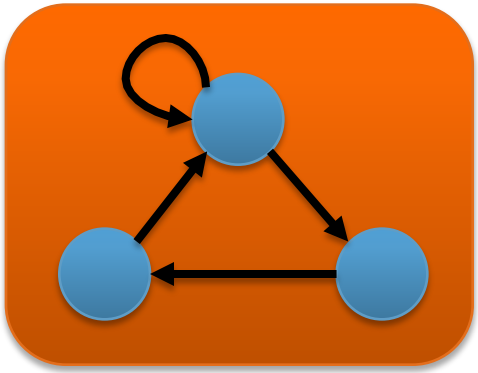
Channels
- Unidirectional connection between two actors
- FiFo semantics
- Operations: *Read():T, Write(T), Peek():T*
- Guards: *mayRead(Int):Boolean, mayWrite(Int):Boolean*

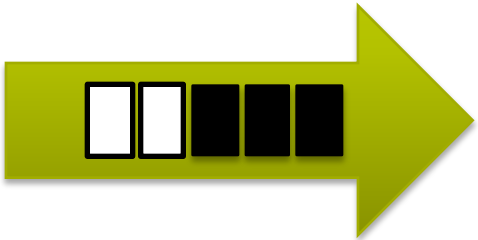


Actors and Channels

Implemented in **ActorX10** framework

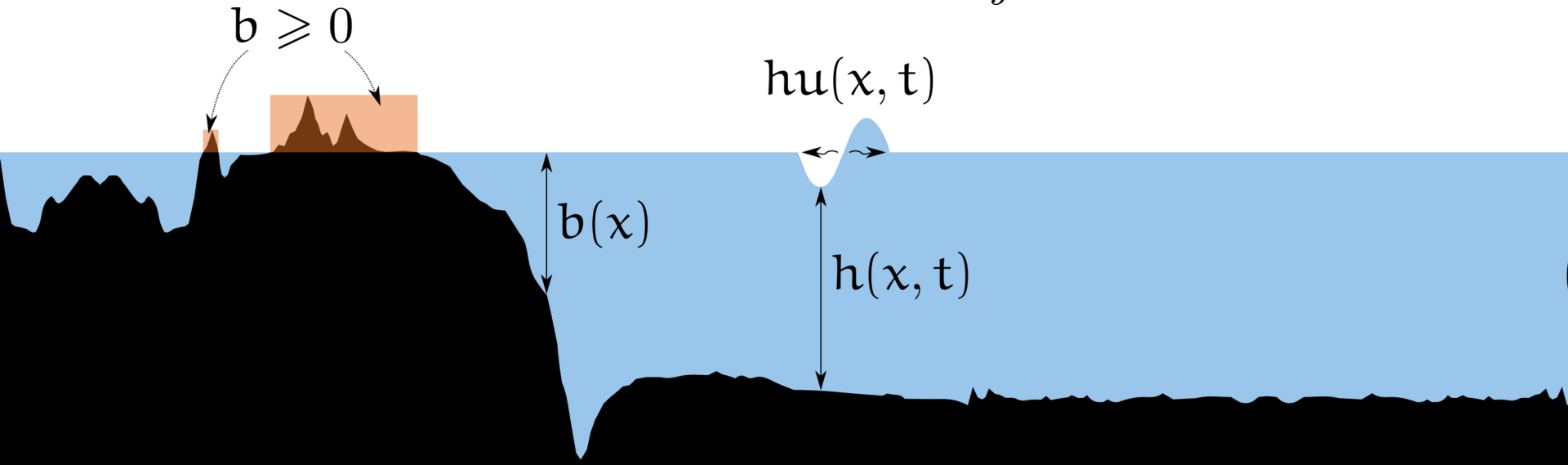
S. Roloff, A. Pöppl, T. Schwarzer, S. Wildermann, M. Bader, M. Glaß, F. Hannig, and J. Teich. **ActorX10: An actor library for X10**. In *Proceedings of the Sixth ACM SIGPLAN X10 Workshop (X10)*. ACM, 2016.





The Shallow Water Equations

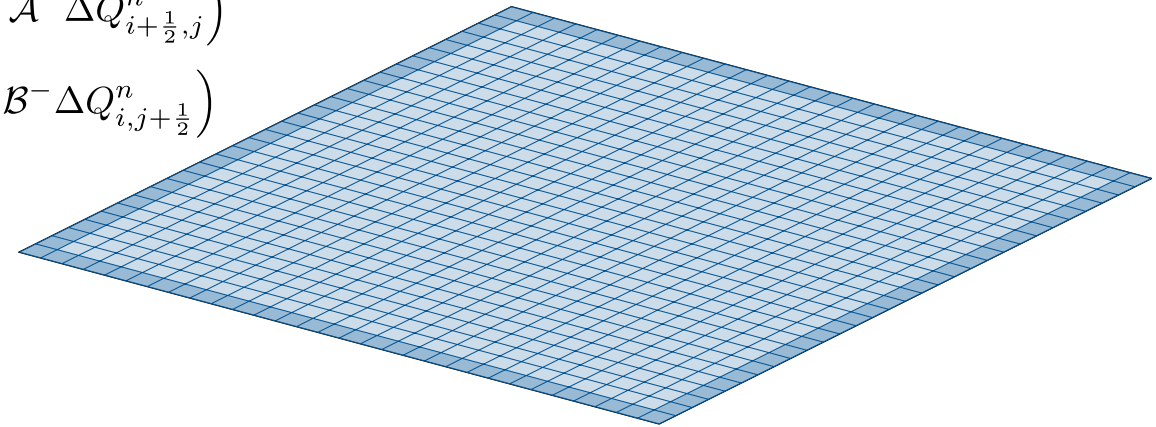
$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = S(t, x, y)$$



Discretization

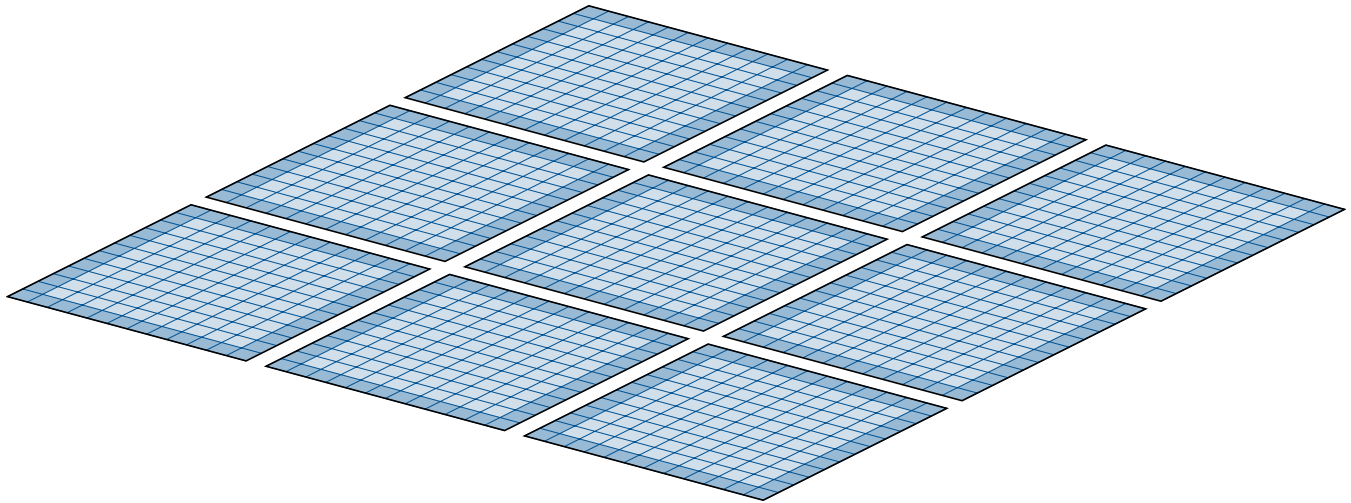
- Continuous simulation domain is discretized onto regular, cartesian grid
- Numerical approach based on LeVeque (R. J. LeVeque, D. L. George, and M. J. Berger. **Tsunami modelling with adaptively refined finite volume methods**. *Acta Numerica*, 20:211–289, 2011)
- Update Scheme:

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



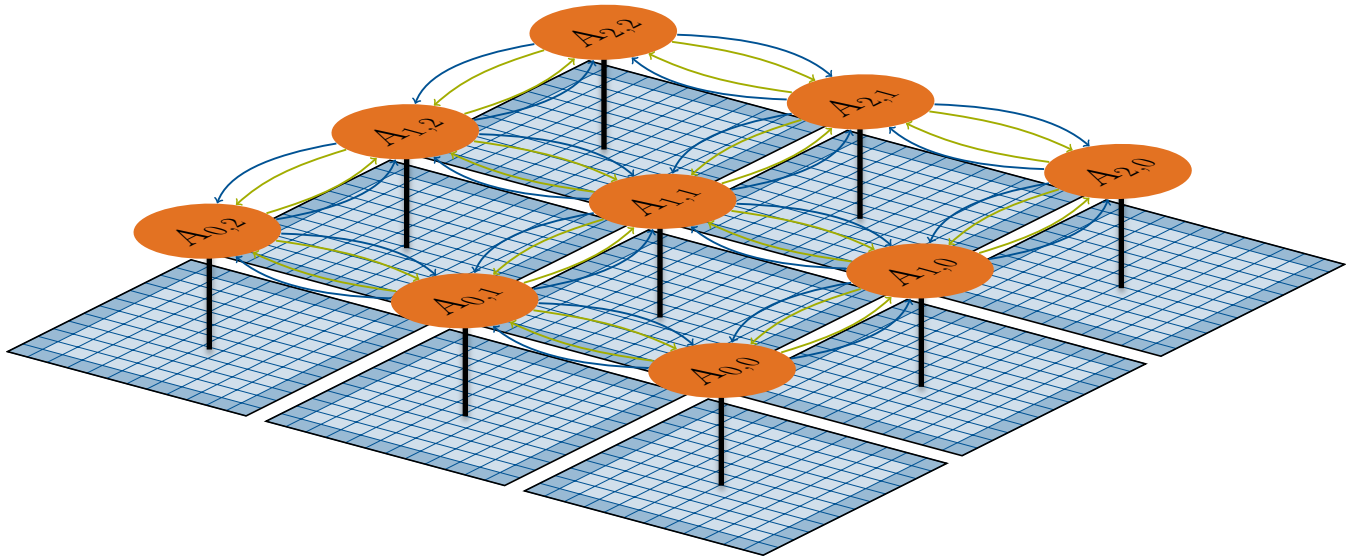
Distribution

Subdivision into rectangular, equally-sized patches with Halo regions



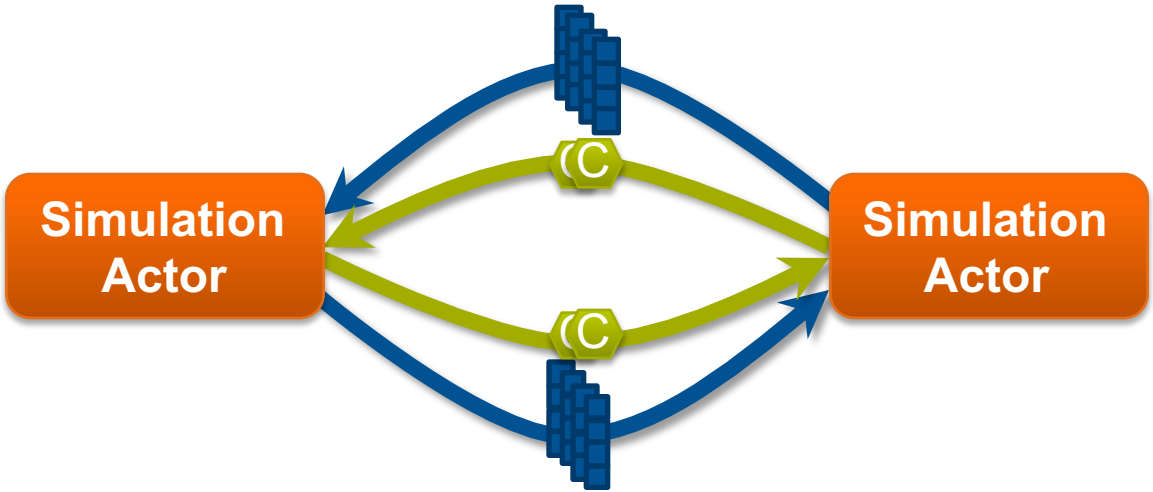
Actor-Based Modelling

- One actor per patch
- Actors connected with direct neighbors



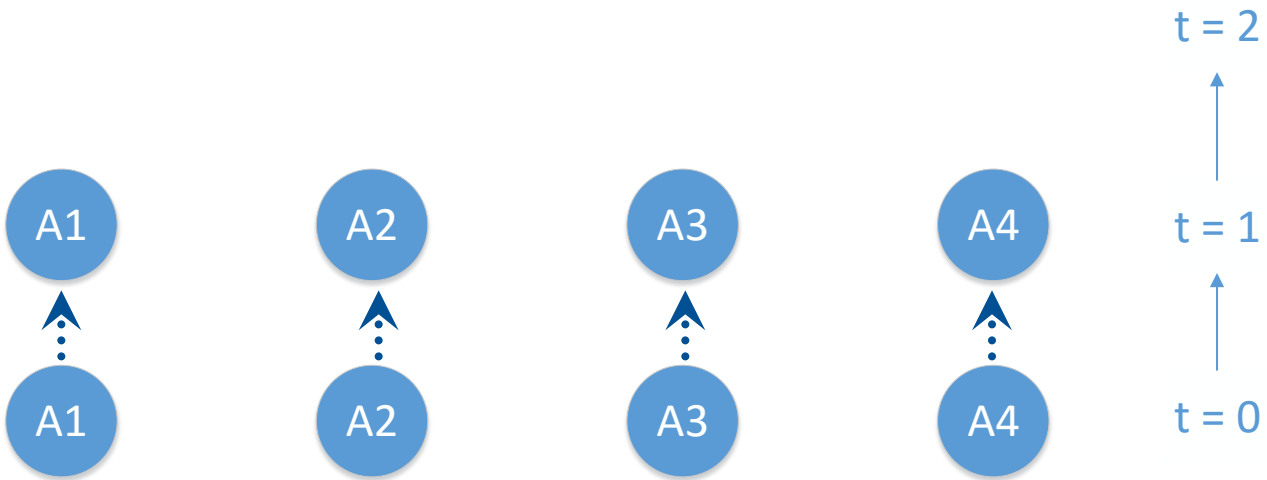
Actor-Based Modelling

Four channels between each actor, one for Simulation Data and one for Control Data in each direction

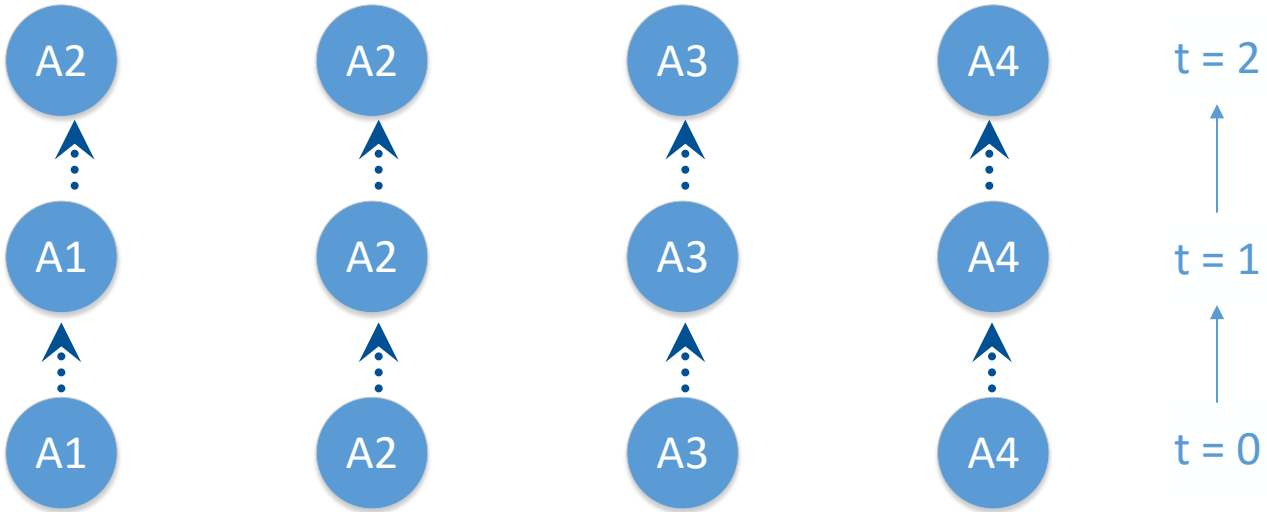


Actor-Based Modelling – Local Coordination

Actor-Based Modelling – Local Coordination



Actor-Based Modelling – Local Coordination

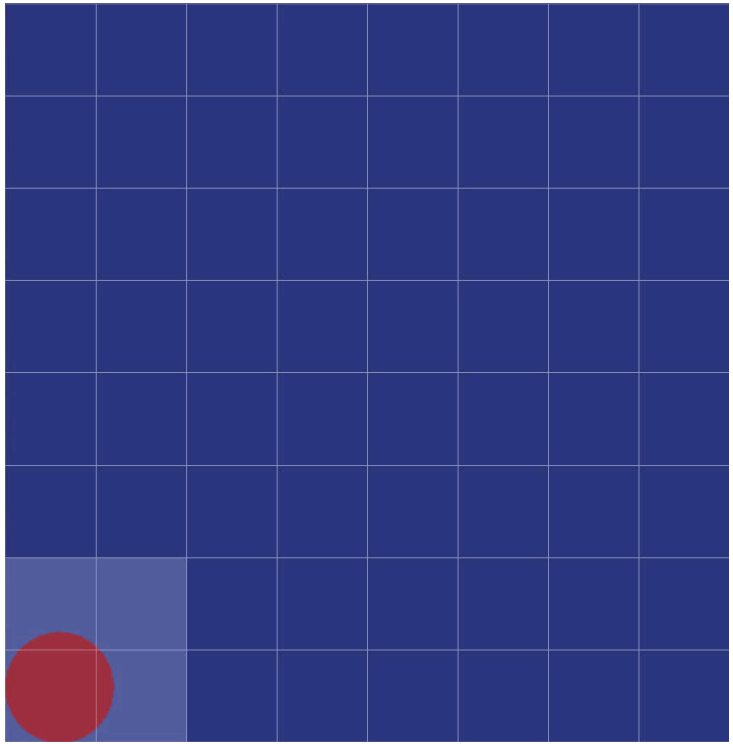


Actor-Based Modelling – Lazy Activation

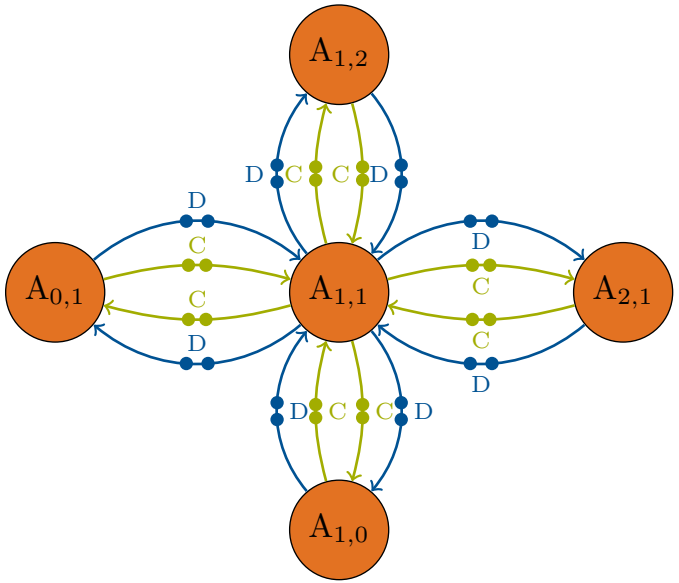
No initial activity in large parts of the simulation domain

- “Lake-at-Rest” is being computed
- Unnecessary use of resources

➢ Just don’t do it!



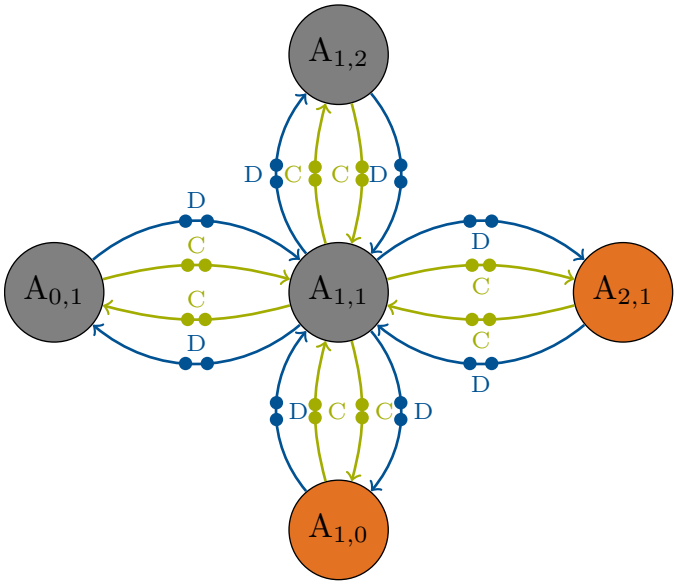
Actor-Based Modelling – Lazy Activation



Actor-Based Modelling – Lazy Activation

Example: $A_{1,0}$ and $A_{2,1}$ initially active, rest inactive.

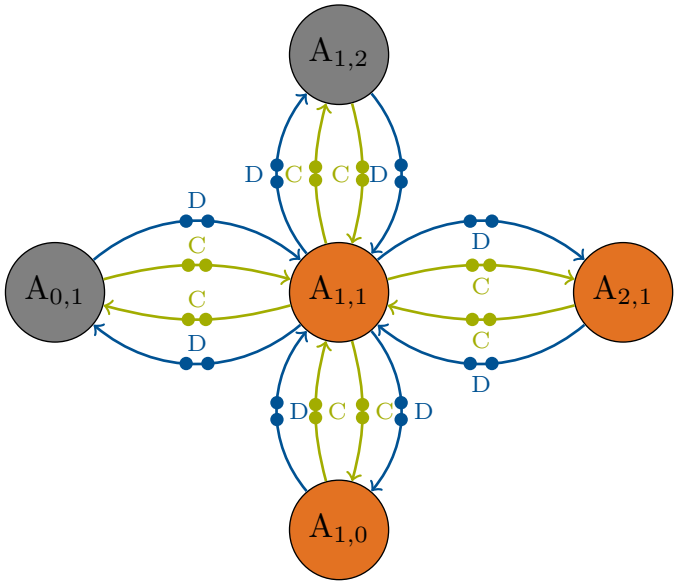
- $A_{1,0}$ just generated first non-zero update to be sent to $A_{1,1}$
- Messages from $A_{1,0}$ to $A_{1,1}$
  - Control Message: “Activate.”
  - Data: Containing the non-zero ghost layer
- $A_{1,1}$ is triggered, sets itself to active
  - Control messages to all neighbors: “I have been activated and will send and receive updates now.”
  - Other active neighbors ($A_{2,1}$) will depend on updates from and send updates to $A_{1,1}$
  - Inactive neighbors are aware of activity of $A_{1,1}$



Actor-Based Modelling – Lazy Activation

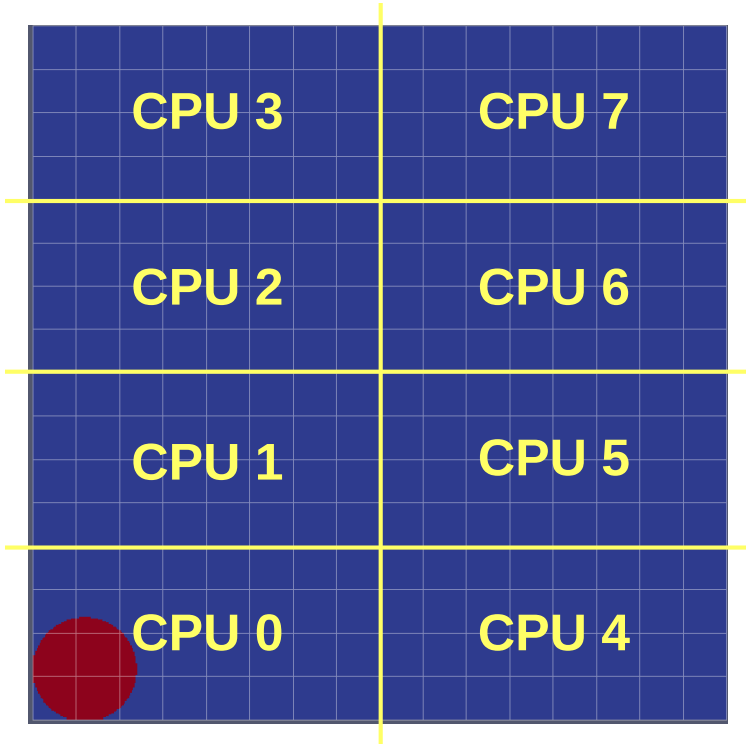
Example: $A_{1,0}$ and $A_{2,1}$ initially active, rest inactive.

- $A_{1,0}$ just generated first non-zero update to be sent to $A_{1,1}$
- Messages from $A_{1,0}$ to $A_{1,1}$
  - Control Message: “Activate.”
  - Data: Containing the non-zero ghost layer
- $A_{1,1}$ is triggered, sets itself to active
  - Control messages to all neighbors: “I have been activated and will send and receive updates now.”
  - Other active neighbors ($A_{2,1}$) will depend on updates from and send updates to $A_{1,1}$
  - Inactive neighbors are aware of activity of $A_{1,1}$



Lazy Activation – Evaluation

- Radial Dam break with initial disturbance in lower left corner of the simulation domain
- 8192×8192 grid cells, divided into patches of size 512×512
- Cluster with 28 dual-socket nodes
  - Intel Xeon E5-2670 CPUs (332.8GFlop/s SP per node)
  - 128GB memory per node (Stream Triad: 108.9GB/s per node)
  - Infiniband QDR interconnect
- 4 patches per CPU core
  - 4 nodes with 2 sockets, 8-core CPUs
  - One X10 Place per socket



Lazy Activation – Evaluation

Evaluation Criterion: CPU time

Without lazy activation:
- Simulation time: 1433s
- CPU time: 12264s (3.41h)

With lazy activation:
- Simulation time: 1203s
- CPU time: 6741s (1.87h)

Simulation-dependent!

Conclusion and Future Work

- Actor-based solver for the shallow water equations
- Initial demonstration of flexibility gained through actor-based modelling
- Expansion of SWE-X10 towards
  - Block-adaptive mesh refinement
  - Local time stepping scheme
  - Load-balancing
- Extend code towards heterogeneous environments
  - Explore resource combinations
  - Find optimal allocations for certain scenarios

Thank You for Your Attention

Acknowledgements

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