Informing coarse-graining through concurrency

Adam Sampson and Jim Bown
White Space Research
University of Abertay Dundee
measure, model → cell
measure, model → cell → replicate
measure, model

replicate

not practical: insufficient computational resources
measure, model → cell
measure, model → cell → region
coarse-grain, simplify
measure, model, \textit{cell} \rightarrow \textit{discard emergent properties?} \rightarrow \textit{region}
measure, model → cell → coarse-grain, simplify → region → validate → replicate

cell  cell  cell
cell  cell  cell
cell  cell  cell
measure, model

cell

cell

cell

cell

cell

cell

cell

cell

cell

cell

cell

cell

cell

cell

cell

coarse-grain, simplify

region

validate

greatest practical scale
cell

measure, model

coarse-grain, simplify

replicate, parallelise

validate

greatest practical scale

cell cell cell

cell cell cell

cell cell cell
Concurrent programming

- Design and implement software in terms of concurrent activities and how they interact
  - Uses include: network servers, robotic control systems, multiplayer games, media processing...
Concurrent programming

- Activities are “lightweight threads”, with their own **state** and **flow of control**
- Modelling entities as concurrent activities means they can **behave** and **develop** independently
  - No artificial ordering on interactions
  - A heterogeneous system, not a homogenised soup
From concurrency to parallelism

- **The runtime system** schedules activities automatically across the available processors
  - ... so it exploits the **natural concurrency** of the system you're modelling to execute in parallel
- Modern concurrent runtime systems – Intel's TBB, the GHC Haskell runtime, CCSP... – have low activity overheads and excellent **scalability**
Smart scheduling

- Scheduling is done while the program is running
  - More information available: better decisions
- Dynamic load-balancing
  - Work stealing finds jobs for idle CPUs
- Informed by interactions between the activities
  - Minimises contention, and improves locality
Distributed simulation

- Making the interactions explicit considerably simplifies **distributing** a problem across a cluster of machines
  - Scalable techniques, minimising **latency** effects
Playing games with space

- Spatial interaction is key to our applications
  - Needs to be dynamic, accurate and fast
- We use tricks developed for real-time collision detection in computer games
Where next?

- In use on a variety of projects (immunology, cell signalling, electricity networks...) using CoSMoS design patterns
- **This summer**: cell physics, blood clotting
- **Longer-term**: cancer modelling in CRISP
  - ... where spatial interaction and heterogeneity are also major concerns
- **Tools**: developing more appropriate interaction mechanisms for simulations – and making the runtime system space-aware
2006, one machine
2006, 18 machines
2011, one machine
region
Thanks to...

- **CoSMoS** (EPSRC)  
  www.cosmos-research.org  
  esp. Paul Andrews, Carl Ritson, Peter Welch

- **CRISP** (SICSA)  
  esp. Jim Bown, Alexey Goltsov, Mark Shovman

Any questions?