Visualization Software and Hardware for In-Silico Brain Research

Stefan Eilemann
Visualization Team Lead
Blue Brain Project, EPFL
Blue Brain Project / Human Brain Project

• BBP: Swiss national research project
  – Launched in 2005
  – Models rodent neocortex at neuronal level
  – Needs about 100TB and 1PFlop/s
• HBP: European FET Flagship
  – Started October 2013
  – Aims to reverse-engineer the human brain
  – Needs about 100PB and 1Exaflop/s (cellular level)
Blue Brain Visualization Infrastructure

• Lausanne:
  – 13 triple-GPU nodes, 12 cores, 24 GB
  – One dual-GPU node, 32 cores, 1 TB, SSD stack
  – 24 MPixel thin-bezel, multitouch display wall
  – 100 inch stereo display

• Lugano (CSCS):
  – 40 dual-GPU nodes, dual Tesla K20
  – Infiniband fabric linked to 4-rack BlueGene/Q
Partners

• Kaust
  – 200 MPixel CAVE, 40 tile display wall, NexCave
  – Collaboration on neuro-glia-vascular modelling and interactive supercomputing

• RWTH Aachen
  – 5x5m AixCave, HBP partner

• UPM
  – 5-sided Cave, Cajal BBP Collaboration, HBP partner
Open Source Software Infrastructure

- DASH: Thread-safe data access and sharing
- CoDASH: Distributed dash (WIP)
- Equalizer: Parallel rendering framework
- Collage: OO network library
- Livre: Scalable OOC volume rendering (WIP)
- DisplayCluster: TACC version + BBP features
• RTNeuron: Parallel simulation renderer¹
• NeuMesh: Mesh generation library
• BBPSDK: BBP data model
• Monsteer: Monitoring and steering (WIP)
• Brion: BBP IO library
• Paraview BBPSDK plugin
  — Custom pipeline based on Zoltan and Thrust¹

¹ Practical rendering of detailed neuron simulations, Hernando et al, EGPGV 2013
Software Infrastructure

- Engineering first, visualization research second
- Tribits-like project quality metrics
  - Unit tests, CR, CI, design review
- Usability
- Reliability
- Ten years or longer lifespan
- Integration with legacy projects
• Interactive Supercomputing
  – Recompute, not store results
• Visualization of multi-scale simulations
  – Moving target due to research nature
• Programming paradigms
• Challenging rendering problems
• Usability of current HPC workflows
• For 10+ years
A Modern HPV Architecture Vision

• Generic, horizontal frameworks
  – Data access and sharing
  – Filter API / Data Model
  – Dataflow library
  – Execution environment
  – Runtime Scheduler

• Filter implementations

• Application logic
Data Access and Sharing

- github:BlueBrain/dash: Boost-style C++ library
- Generic **data access**
  - Directed acyclic graph with ‘any’ Attribute data
  - Per-thread Context ‘memory space’
  - Copy on write semantic, commit propagation
- Fast and safe data **sharing** between threads
- Wait-free reads, lock-free writes
- External data distribution (CoDASH prototype)
Filter API and Data Model

- Introspectable
- Multiple independent dash inputs and outputs
- Optional data model (EAVL?)
  - Data parallel execution
  - Scheduling
- Constraints (GPU, Memory, ...)
- Encapsulates algorithms
Filter Library

- Construction, serialization and storage of dataflow graphs
- Fan-in, fan-out
- Multiple consumers and providers
- Synchronization policies
Execution Environment

• Distributed service architecture
  – Resource and filter discovery
  – Runtime linking
  – Failure detection and handling
• Push-based execution
  – Multithreaded and multiprocess
  – Task-parallel and data-parallel
  – DASH and CoDASH do most of the work
• BBP/HBP to address in about five years
• Meta “OS”
• Resource allocation and scheduling
• Multi-user and multi-application
• Hot research area (Data locality, Cloud, ...)
• Likely shim between execution environment and third-party software(s)
• HPC is out-innovated by large data computing
  – Hard to find new talent
  – Critical mass for software ecosystem
  – Funding reallocated to big data
• HPC needs to modernize
  – Tools, usability, reliability, software engineering, …
  – Focus on results, not bit/flops counting
• Simulation-based brain research is still young
Large Scale Visualization Future

- HPV: Take the best of HPC and large data
  - Usability, robustness, modularity, engineering
  - Scale and efficiency
  - Data stores and query languages

- Engage community
  - Open source generic frameworks
  - Domain-specific applications
Let’s do it!

Stefan.Eilemann@epfl.ch

BTW: We’re hiring!
• **Steering**
  – Wrap simulation as a filter
  – Change filter inputs

• **Data staging**
  – Expose DHT as a filter with inputs and outputs

• **Display Walls**
  – Input pixels, output events/state
• Context: data view – ‘address space’
• Node: DAG element
• Attribute: any data
• Commit: change set
• Wait-free reads
• Fast writes
• No data model