ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

DASH: Data Access and Sharing

Smart Distributed Memory and Synchronization for Multi-Threaded Applications

Stefan Eilemann

Blue Brain Project (BBP), École Polytechnique Fédérale de Lausanne (EPFL)

Blue Brain Project

DASH: Motivation, API & Implementation

Hardware Evolution

- Parallelism due to clock rate wall
- Less memory and bandwidth per instruction

Drives

Software Parallelism

- Need for thread-safe data access
- Data and task parallelism

Creates

Software Challenges

- Missing thread-safe library support
- Trade-off between locking and copying

Require

Modern Library

- Generic data access
- Multi-threaded task-parallel algorithms

Results in

DASH

Efficient Multi-Threaded Data Access, Sharing and Synchronization

Simple Generic Data Storage Library for any C++ type

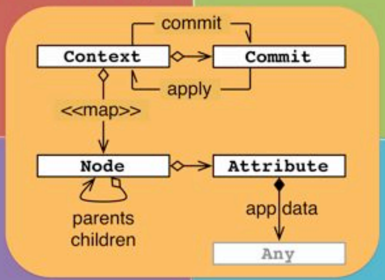
Open Source C++ Library

Context

Provides isolated view on stored data
Per-thread "distributed memory"

Commit

Set of changes emitted by and applied to context



Node

Has attributes, parent and child Nodes
Forms DAGs of application data

Attribute

Holds "any" application C++ object
Objects implement serialization

Implementation

Based On

- Atomic variables
- Lock-free algorithms
- boost::any for application data

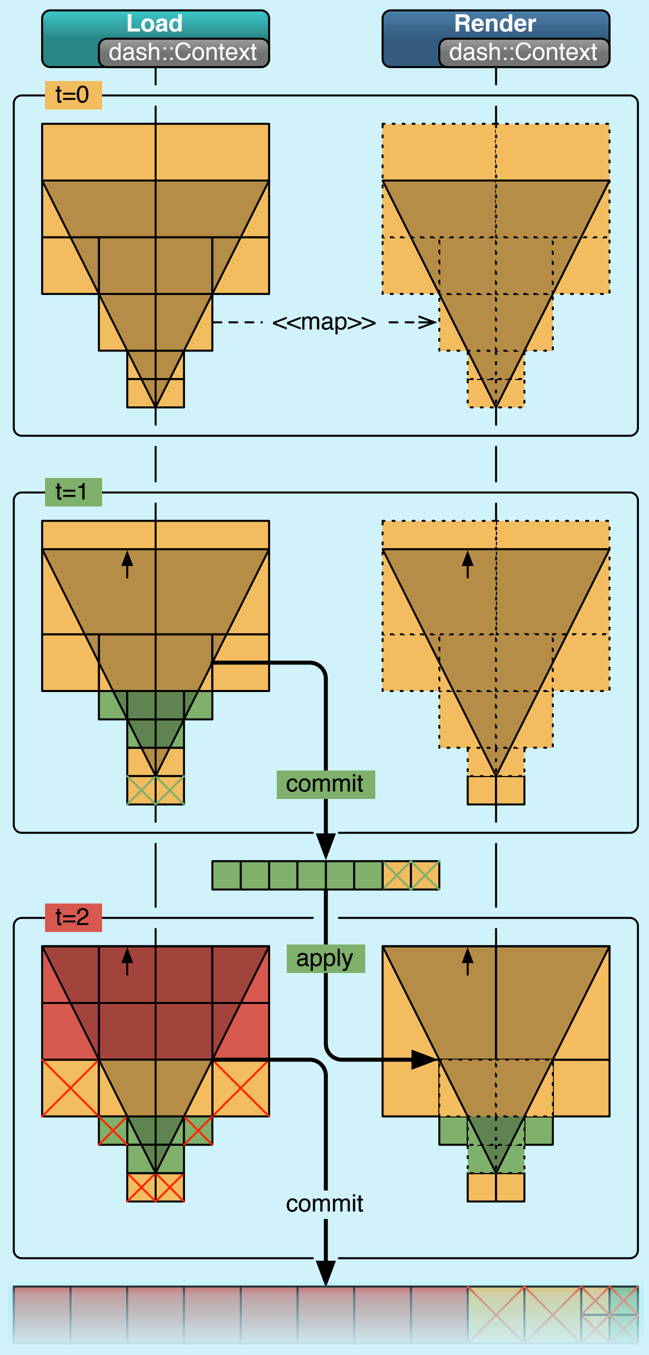
Guarantees

- Lock-free & wait-free concurrent read accesses
- Fast writes with no data contention
- No copies for data updates between contexts

Provides

- Memory efficiency: copy on write
- Fast data access and thread synchronization
- Extensibility to persistency and data distribution

Use Case: Asynchronous LOD Loading



Render **Load**

Each thread has **dash::Context**

Both contexts initially **share** all data

dash::Nodes form graph

LOD data stored in **dash::Attributes**

Graph is mapped to both contexts

Data is copied on write (**COW**)

Load

Update data for new frustum

Updated nodes are **copied**

Unchanged nodes are shared

Commit bundles all changes

Render

Context provides **unchanged view**

Load

Create **commit** for next t=2

Render

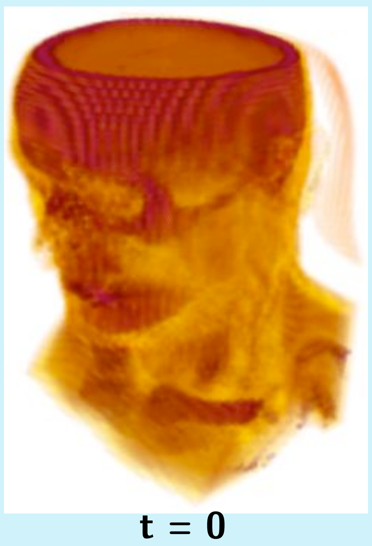
Apply commit from loader thread

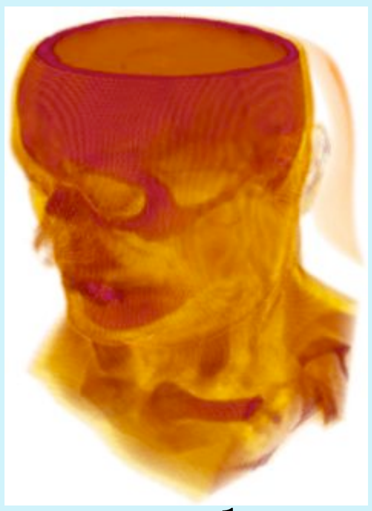
Replaced nodes are unreferenced

Context provides view at t=1

▽ View Frustum | 🟧 LOD Node at t=0 | 🟩 Shared LOD Node at t=1 | ☒ Deleted LOD Node at t=2 | 🟥 Added LOD Node at t=2

Real-time Volume Rendering for the Visible Male Dataset*


t = 0


t = 1

t = 2

bluebrain.epfl.ch

https://github.com/BlueBrain/dash

* Dataset courtesy of NVIDIA, GPU Gems.