MUSIC
the Multi-Simulation Coordinator

Örjan Ekeberg and Mikael Djurfeldt
CSC, KTH
Introduction

The purpose of MUSIC

- On-line pre- or post-processing of huge amounts of data for a parallel simulator within the cluster
- Connect models developed for different parallel simulators
- Promote re-usability through modularity
mpirun -np 4 nrniv my_simulation.hoc
mpirun -np 3 -hostfile h1 my_lgn_model
mpirun -np 5 -hostfile h2 nrniv my_simulation.hoc
mpirun -np 3 -hostfile h3 nest my_simulation.sli
Introduction
Introduction

Recommendation from the report of the 1st INCF Workshop on Large-scale Modeling of the Nervous System:

“Implement an experimental framework for connecting software components. A feasibility study should be performed regarding the possibility of on-line communication between different software modules, for example two parallel simulators. INCF should allocate resources for implementing a software library with a communication interface.”

- MUSIC standard and software provided and supported by the International Neuroinformatics Coordinating Facility (INCF)
- Being developed by the CSC, KTH in a collaborative partnership with the INCF
- Released publicly under the GPL license through the INCF Software Center in early 2009
Introduction

Design Goals

- Portability
  Based on C++ and MPI

- Simplicity
  Minimal impact on existing simulators

- Independence
  Encourage independent tool development

- Performance
  High bandwidth, low latency through use of MPI

- Extensibility
  Some classes in API can be subclassed
Users View of MUSIC

A multi-simulation where several parallel applications exchange runtime data.
Users View of MUSIC

- MUSIC-adapted applications present **Ports**
- Ports have **names**
- User **connects** ports via a *configuration file*
**Users View of MUSIC**

Typical configuration file `my_simulation.music`:

```plaintext
stop_time = 1.0
[model]
  binary = nrniv
  args = my_simulation.hoc
  np = 40

[EEG]
  binary = eegsynthesizer
  args = geometry.dat 50e-6 10
  np = 12
  model.eneurons -> input [1600]
```

```bash
mpirun -np 52 music my_simulation.music
```
Programmers View of MUSIC

Execution goes through three phases

- **Launch phase**
  Outside the control of the application

- **Setup phase**
  Declaration and mapping of ports

- **Runtime phase**
  Simulation and transfer of data
Programmers View of MUSIC

Each application is responsible for:

1. Initializing MUSIC
2. Creating Ports
3. Mapping Ports
4. Initiating the Runtime Phase
5. Advancing Simulation Time
Programmers View of MUSIC

Initializing MUSIC

```c
int main (int argc, char *argv[])
{
    setup = MUSIC::setup (argc, argv);
    comm = setup->communicator ();
    ...
}
```
Programmers View of MUSIC

Programs announce willingness to send or receive data via ports.
Creating and mapping a port

```
p = setup->publish_cont_output ("out");

array_data m (state_vars, MPI::DOUBLE, mybase, mysize);
p -> map (&m);
```
Programmers View of MUSIC

Supported Data Types

- **Continuous** — Time varying values
  
  Sender: Reading from user data structures
  Receiver: Writing into user data structures

- **Events** — Spikes
  
  Sender: User calls an insertion function
  Receiver: MUSIC calls user-supplied handler

- **Messages** — Arbitrary strings of bytes
  
  Sender: User calls an insertion function
  Receiver: MUSIC calls user-supplied handler
Programmers View of MUSIC

Initiating the runtime phase

```cpp
... runtime = new MUSIC::runtime (setup, 0.0001);

while (runtime->time () < stoptime)
{
    ... runtime->tick ();
    ...
}
```
Implementers View of MUSIC

What goes on behind the scene?

- **Spatial Aliasing**
  Data resides on different processors

- **Temporal Aliasing**
  Different applications may use different time steps
Implementers View of MUSIC

Spatial Aliasing

Diagram showing the interaction between sender and receiver with distributed data and width.
Implementers View of MUSIC

Temporal Aliasing

Sender

\[ s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \]

MPI

MUSIC

Receiver

\[ r_1 \rightarrow r_2 \]

Simulated Time
Implementers View of MUSIC

Temporal Aliasing

Sender

MPI

MUSIC

Receiver

Simulated Time

s1 s2

r1 r2 r3 r4 r5
Request For Comments

- MUSIC project page
  http://www.incf.org/programs/modeling/music-multi-simulation-coordinator

- RFC