MUSIC
the Multi-Simulation Coordinator

Mikael Djurfeldt, INCF/KTH and Örjan Ekeberg, KTH
Introduction

What is MUSIC?

- An API allowing large scale neuron simulators which use MPI internally to exchange data during runtime
- An implementation of a C++ library providing this API
Introduction

2.6 Tb/s 2.6 Gb/s Simulator EEG
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

The purpose of MUSIC:

Allow multiple applications to run together and communicate within the parallel computer

▶ 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
▶ Run multiple instances of the same simulator on different parameter sets in one parallel job
▶ Promote re-usability through modularity
MUSIC - an INCF project

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)
- 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
Users View of MUSIC

What is communicated?

- **Continuous** — Time varying values
- **Events** — Spikes
- **Messages** — Arbitrary strings of bytes

Communicated through named ports presented by the application
Users View of MUSIC

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

Typical configuration file my_simulation.music:

```
stoptime=1.0

[A]
    binary=my_lgn_model
    np=5
    out → B.in
    out → C.in

[B]
    binary=nest
    args=...
    np=3
    to_L5_pyramidal → C.in
    from_L5_pyramidal ← C.out

[C]
    binary=nrniv
    args=my_simulation.hoc
    np=3
```

```
mpirun -np 11 music my_simulation.music
```
Spatial Aliasing—different distribution of data

![Diagram showing distributed sender and receiver data with width indicated.](image-url)
Temporal Aliasing—different time steps

Case 1: Sender ticking faster than receiver
Temporal Aliasing—different time steps

Case 2: Sender ticking slower than receiver

![Diagram showing temporal aliasing with different time steps for sender and receiver.]

- **Sender**:
  - $s_1$
  - $s_2$

- **MPI**

- **MUSIC**
  - $r_1$
  - $r_2$
  - $r_3$
  - $r_4$
  - $r_5$

- **Receiver**

Simulated Time: $s1$ $s2$ $r1$ $r2$ $r3$ $r4$ $r5$
Developers 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
Developers 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
Developers View of MUSIC

Initializing MUSIC

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

Programs announce willingness to send or receive data via ports
Developers View of MUSIC

Creating and mapping a port

```cpp
p = setup->publishContOutput ("out");

ArrayData m (state_vars, MPI::DOUBLE,
            mybase, mysize);

p -> map (&m);
```
Developers 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
Developers View of MUSIC

Initiating the runtime phase

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

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

Cortex outputs

5 MS neurons

5 FS neurons

Index

0 0.2 0.4 0.6 0.8 1

Time (s)

50 100 150 200 250 300 350 400 450 500

0.846 s

0.846 s
Benchmark

n(i,x): number of music channels per population
mop: music_out_proxy
mip: music_in_proxy
k: number of connections from/to population
pn: parrot_neuron
Benchmark

A

B

number of cores

simulation time / biological time

4 8 16 24 32 40 48

4 8 16 32

number of cores

simulation time / biological time

4 8 16 32

4 8 16 32
Status

- Released in the beginning of March, 2009
- Downloadable at the INCF Software Center
  http://software.incf.org
- Supported by the INCF
- Users manual
Status

Music interfaces:

▶ NEST — Moritz Helias and Jochen Eppler

▶ MOOSE — Niraj Dudani and Johannes Hjorth

▶ Neuron — Michael Hines
MUSIC
the Multi-Simulation Coordinator

Mikael Djurfeldt, INCF/KTH and Örjan Ekeberg, KTH