An Hybrid MPI/Stream Programming Model for Heterogeneous High Performance Systems

Emilio P. Mancini, Gregory Marsh, Dhabaleswar K. Panda
{mancini, marshgr, panda}@cse.ohio-state.edu

May 19th, 2009
Outline

• Introduction

• The MPI, Stream and Hybrid programming models

• Architecture of the hybrid framework

• Case study: a financial application

• Conclusions
Motivation

• The increased number of nodes in modern computational systems introduces **implicit heterogeneity**:  
  – For example, they can use different levels of switches

• It is difficult to use different computational cluster at the same time with the same parallel program (**clusters of clusters**)

• We want to study a way to exploit the locality in computational clusters, and in clusters of clusters
Motivation

• Two connected clusters have different latencies and bandwidths (depending on the source and the destination):
  – Inter-core
  – Inter socket
  – Level of the switch
  – Inter cluster
Motivation and problem statement

• MPI proposes a flat computational model that can well exploit the locality

• The stream models, with the light coupling between computational units: it is useful for heterogeneous networks

• This paper we study how to integrate MPI and Stream programming models in order to exploit network locality and topology

• In this paper we present a framework which integrates the two models.
Outline

• Introduction

• The MPI, Stream and Hybrid programming models

• Architecture of the hybrid framework

• Case study: a financial application

• Conclusions
MPI and Stream models concepts

• An MPI program is a set of autonomous processes that exchange data through message passing

• A stream can be described as an unbounded set of data
• A series of “kernels” process each element of the stream
• The Kernel’s inputs and outputs are streams
Stream model

- The data model is modeled as **transient data streams** (not persistent relations)

- They arrive continuously in **unpredictable** way, and in **unbound streams**

**siblings -> MPI application**
The hybrid model

- In the hybrid model a subset of kernels can be mapped on a set of MPI processes;
- Or, from another point of view, a set of MPI processes can be transformed as a stream kernel.
Outline

• Introduction

• The MPI, Stream and Hybrid programming models

  • Architecture of the hybrid framework

• Case study: a financial application

• Conclusions
The launching process

• The launcher requires a description of the whole system, and synchronization of MPI and sequential kernels in different nodes

• An XML file describes the task graph

• From the XML file the launcher dynamically produces MPI hostfiles and startup scripts

• At launch time, every kernel registers itself with the middleware or polls the stream autonomously
The hybrid framework sequence diagram

Stream Launcher
MPI Launcher
MPI Kernel
Streams middleware
Sequential kernel

Run
Run
Run

Streams middleware
Open stream
Open stream

Parallel
Put
Get
Close stream

End
End

Complete MPI application
The hybrid framework architecture

- The launcher starts both stream kernels and MPI applications
- The core is a modularized communication API
- A common interface allows to interact with different underlying protocols
- The graph management module builds a view of the application tasks
A simple example

The main function registers the kernels, and starts the streams

```c
int main (...) {
    osf_KernelContext_t *kctx;
    osf_Init(...);
    osf_RegisterKernel(0, OSF_KRNTYP_POLLING,
                        OSF_KRN_STATELESS, SourceKernel, &kctx);
    osf_RegisterKernel(1, OSF_KRNTYP_POLLING,
                        OSF_KRN_STATELESS, FilterKernel, &kctx);
    osf_StartStreams();
    osf_Finalize();
}
```
A simple example

A source kernel puts new data into a stream:

```c
osf_Result_t SourceKernel(osf_KernelContext_t *ctx) {
    static osf_Stream_t *s = NULL;
    if (s==NULL)
        osf_Open(&s, OSF_STR_OUT, 1);
    ...
    record = sin(t)+sin(4+2*t);
    osf_Put(s, &record, sizeof(record) );
    return OSF_ERR_SUCCESS;
}
```
A simple example

A filter kernel gets the data from a stream, elaborates them, and eventually, puts them in another stream.

```c
osf_Result_t FilterKernel(osf_KernelContext_t *ctx) {
    static osf_Stream_t *sIn = NULL;
    if (sIn == NULL)
        osf_Open(&sIn, OSF_STR_INPUT, 0);
    ...
    res = osf_Get( sIn, &x, sizeof(double), &receiv );
    return OSF_ERR_SUCCESS;
}
```
Outline

• Introduction

• The MPI, Stream and Hybrid programming models

• Architecture of the hybrid framework

• Case study: a financial application

• Conclusions
Case study: a financial simulation application

- The application simulates a stock
- It generates random orders
- A matching engine compares offers, bids and quantities
- When it elaborates an order, it sends a confirmation

max_bid: Highest offered buying price. All orders at and below this price are bid type.

min_ask: Lowest offered selling price. All orders at and above this price are ask type.

<table>
<thead>
<tr>
<th>0</th>
<th>......</th>
<th>450</th>
<th>451</th>
<th>452</th>
<th>453</th>
<th>..........</th>
<th>546</th>
<th>547</th>
<th>548</th>
<th>549</th>
<th>......</th>
<th>999</th>
</tr>
</thead>
</table>

Double-ended queues with pending orders at that price level.

orders in

orders out

orders in

orders out
Case study: a financial simulation application

- The application has 4 tasks
- The hybrid version uses both MPI and Stream primitive to communicate
- The stream kernels are not synchronized with one another
Case study: a financial simulation application

- The experiment was lead on a single cluster
- The hybrid version shows a better execution time
- The improvement varies from 5% to 32% (simulating 30,000 orders/s)
Conclusions and future directions

• We proposed a way to exploit the locality using a hybrid Stream/MPI programming model

• We presented the prototype of a hybrid framework, and validated it using a financial simulation

• We plan to experiment this approach using clusters of clusters

• We plan to integrate the framework in the message queue of MPI middlewares
Thank you

An Hybrid MPI/Stream Programming Model for Heterogeneous High Performance Systems

{mancini, marshgr, panda}@cse.ohio-state.edu