INTERPROCESS DATA OBJECT COMMUNICATION

Roberto A. Vitillo (LBNL)

CONTEXT

- AthenaMP communications
  - reader process sends events to workers
- Coprocessor communications
  - Athena[MP] jobs interacting with a GPU server process
- Available IPC mechanisms
  - shared memory with explicit synchronization
  - message passing with implicit synchronization

MESSAGE PASSING MODEL

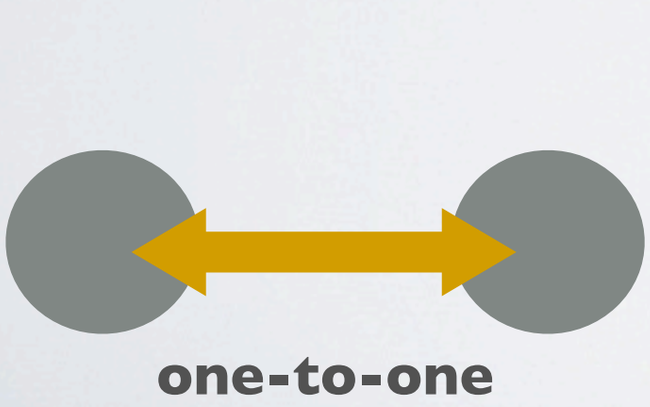
- One of the most successful models for providing concurrency
  - data and synchronization in a single unit
- Actor Model
  - processes have an identity
  - communicate by sending messages to mailing addresses
  - Erlang, Scala
- Process calculi
  - processes are anonymous
  - communicate by sending messages through named channels
  - Go Programming Language

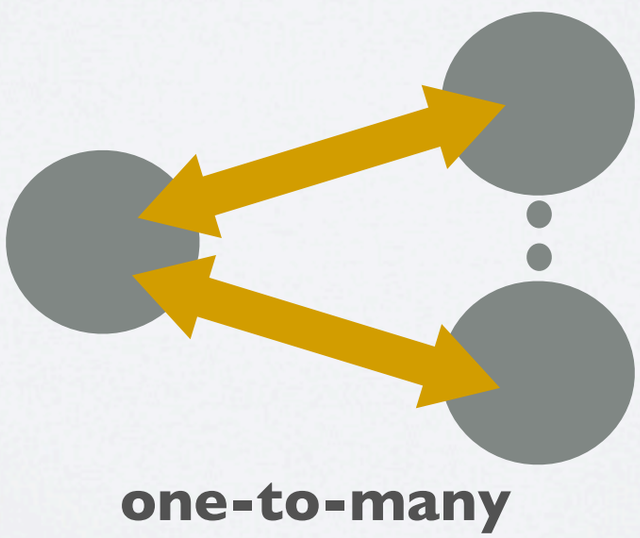
PATTERNS

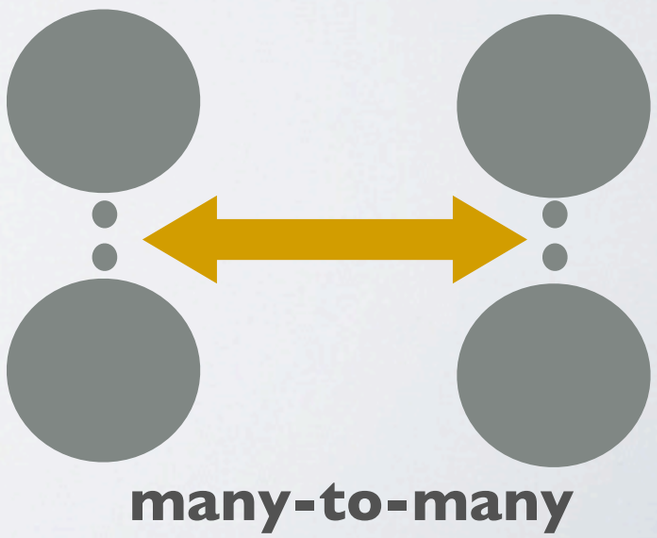
- Producer & Consumer
  - producer pushes messages
  - consumer pulls messages
- Client & Server
  - client makes a request
  - server replies to a request

CHANNELS

- Properties of a channels:
  - name
  - context (thread, local-process, distributed-process)
  - asynchronous(k)
  - topology





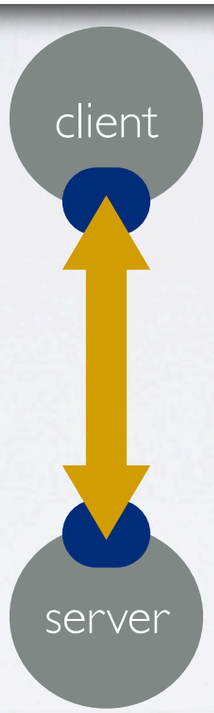


SOCKETS

- Each end of a channel is attached to a Socket
- Different patterns have different Sockets,
  - e.g. ProducerSocket, ConsumerSocket
- A Socket allows to:
  - send() buffers of data to its peers (buffer-blocking)
  - receive() buffers of data from its peers (blocking)

SOCKETS (2)

```
Channel channel("service", ONE_TO_ONE)
ISocket *socket = factory->createClientSocket(channel);

socket->send("ping", 5);
socket->receive(&buffer);
```

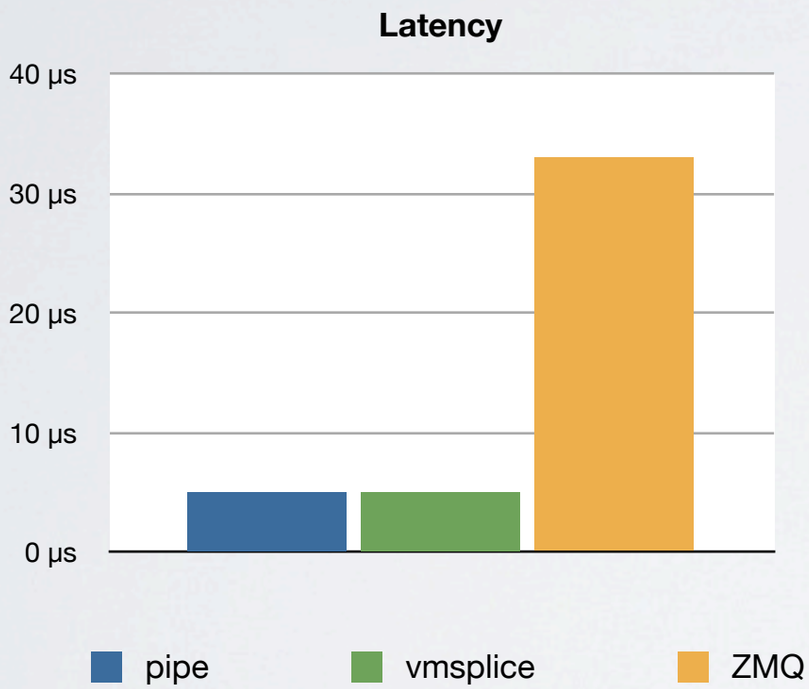


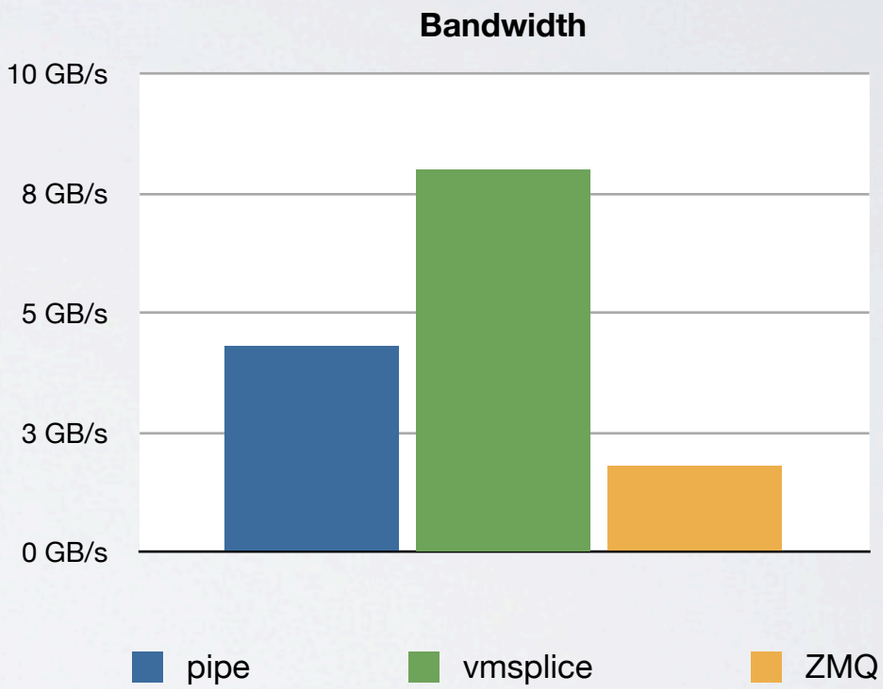

```
Channel channel("service", ONE_TO_ONE);
ISocket *socket = factory->createServerSocket(channel);

while(true){
    socket->receive(&buffer);
    socket->send("pong");
}
```

IMPLEMENTATION

- The API is currently implemented with ZeroMQ
  - provides a default fall back implementation
  - lock-free queues for threads
  - AF_UNIX sockets for local processes
  - TCP sockets for distributed processes
- The implementation switches according to the channel configuration
  - E.g. one-to-one, producer-consumer uses a UNIX pipe with vmsplice()

IMPLEMENTATION (2)





- One-to-one, client-server
- Kernel 3.5, Ivy Bridge
- ZMQ 2.2

CONCLUSION

- Library provides an uniform message-passing abstraction for inter-process communication
- Data and synchronization in a single unit
- Communication patterns and topologies allow to
  - reduce latency
  - increase bandwidth
  - express parallelism
- More implementations will be considered
  - MPI?

QUESTIONS?