

Performance Evaluation of GEMINI

Presented by Matthew Gagne

Queen's University, Kingston, Ontario (Canada)

Student from IAESTE local committee at AGH, Krakow

Performance Evaluation of GEMINI

Outline:

- GEMINI Overview
- Communication Channels
- Metrics/Methodology
- Testing & Results
 - Throughput/Latency
 - Sustained event submission
 - Subscription
 - Query response
 - Overhead
 - Delayed event submission
- Future Testing
 - Scalability
- Conclusion: Questions/Comments

Performance Evaluation of GEMINI

Goals:

- Develop a set of performance metrics to evaluate GEMINI
- Decide on an appropriate testing methodology
- Evaluate components of GEMINI using established metrics/methodology

GEMINI Overview

Monitor

- Runs as a web service in Globus
- Collects information from active sensors

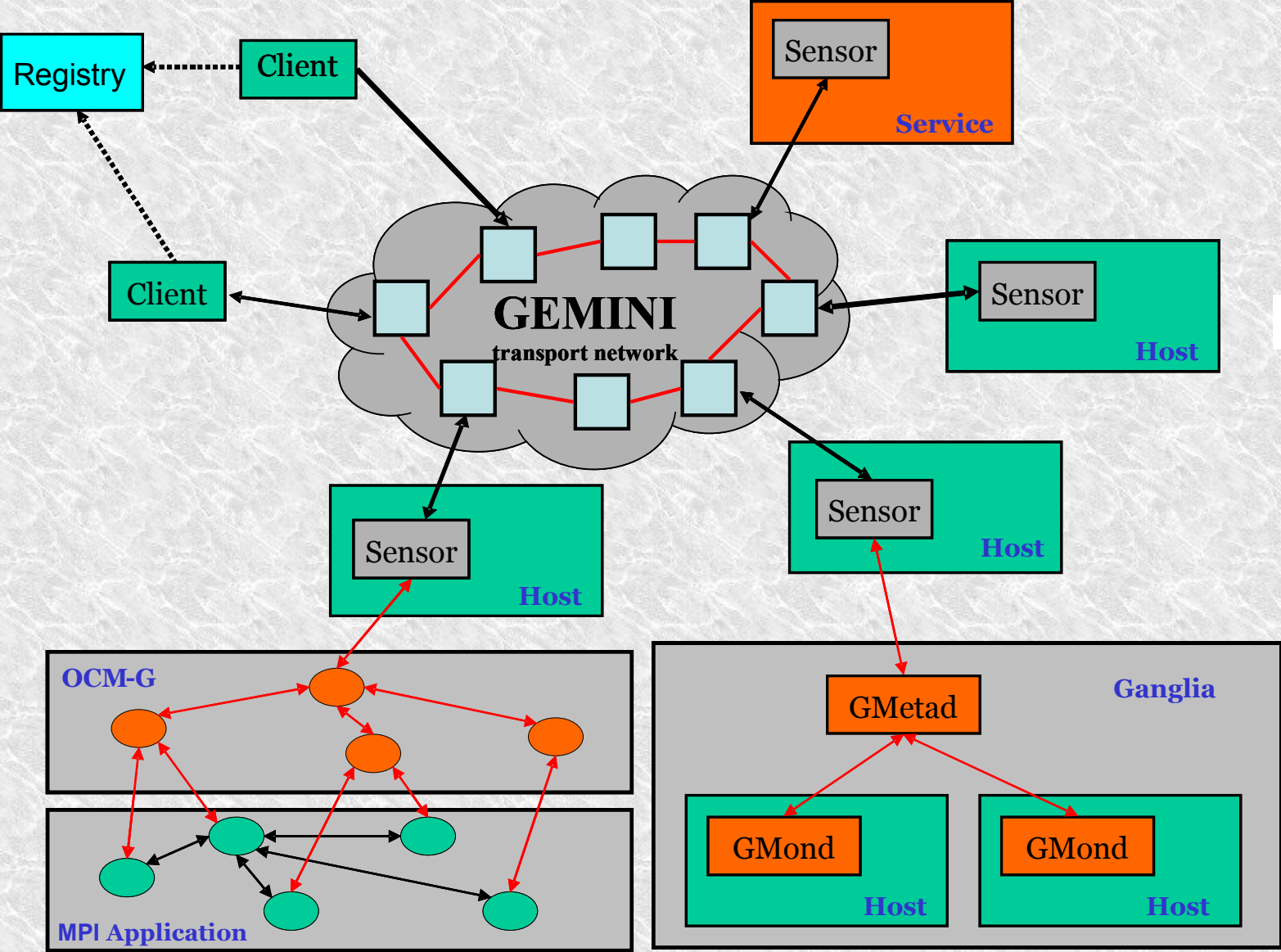
Sensor

- Implemented in workflow application, or separately

Clients

- Query - Pulls stored data from sensor, once per query
- Subscribe - Continuously displays received data from sensor as it is published/pushed

GEMINI Overview



Communication Channels

Internet Communications Engine (ICE)

- Fast XML based middleware for remote procedure calls
- Used for communication in GEMINI between clients, monitors, and sensors.
- Implemented in C++ and Java in GEMINI

Remote Method Invocation (RMI)

- Implemented in Java for communication between instrumented applications and sensors in GEMINI

Test Machine Setup

VMWare virtual machine running on host:

- AMD Athlon MP 2.0 GHz
- 128MB RAM
- Gentoo Linux

Software running on host machine:

- Globus (GT4) (globus-start-container -nosec)
- ICE 2.1.2 (icebox)
- GEMINI (gemini-sensor, gemini-query, gemini-subscribe)

Performance Metrics

Latency

- Delay for GEMINI to respond to certain data requests/submissions
- Throughput
- The amount of data that can be processed by GEMINI over a period of time

Scalability

- Ability of GEMINI to handle growing number of clients/sensors/applications with acceptable performance decrease

Methodology – Testing Process

Boot virtual machine to command prompt

Initialize monitoring infrastructure

- Icebox
- Globus (incl. GEMINI Monitoring Service)
- RMI Registry
- GEMINI sensor

Run test scripts

Collect results

Reboot virtual machine for next test

Latency/Throughput

Tests:

- Sustained event submission
- Delayed event submission
- Event subscription
- Query response time

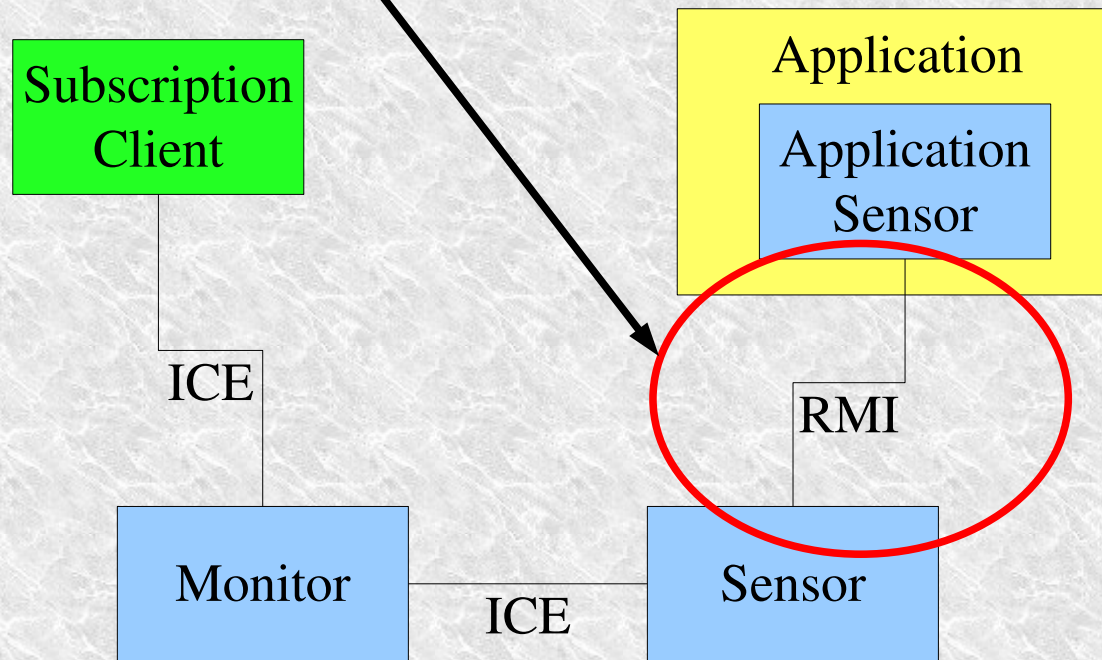
Latency/Throughput - Sustained event submission

Sustained event submission test:

- Create an instrumented workflow application that sends as many events as possible to designated test sensor.
- Insert timing statements before and after event creation and submission
- Run 1000 event submissions, wait, repeat for 10 runs

Latency/Throughput - Sustained event submission

Evaluation of RMI communication channel between application sensor and sensor



Latency/Throughput - Sustained event submission

Sustained event submission test (code example)

```
for (int r = 0; r < runs; r++) {  
    createEventTime = 0;  
    sendEventTime = 0;  
    for (int i = 0; i < iterations; i++) {  
        time = System.currentTimeMillis();  
        event = WorkflowSensor.createEvent("Test event " + i);  
        createEventTime += System.currentTimeMillis() - time;  
  
        time = System.currentTimeMillis();  
        WorkflowSensor.sendEvent(event);  
        sendEventTime += System.currentTimeMillis() - time;  
    }  
    //store times in array for each run  
}
```

Latency/Throughput - Sustained event submission

Results for 10 runs, 1000 submissions per run:

Run #	Total Send Time (sec)	Total Create Time (sec)	Total Event Time (sec)
1	22.496	3.609	26.105
2	20.733	2.180	22.913
3	21.826	1.843	23.669
4	21.460	1.777	23.237
5	19.208	1.753	20.961
6	21.644	1.811	23.455
7	23.525	1.790	25.315
8	21.782	1.769	23.551
9	20.814	1.783	22.597
10	23.085	1.781	24.866
Average	21.657	2.010	23.667
Std Dev	1.242	0.576	1.467

Time/event (msec)

24

Latency - Delayed event submission

Delayed event submission test:

- Create an instrumented workflow application that contains 4 event submissions, separated by a time interval of 500 milliseconds.
- Insert timing statements before and after all 4 events have been submitted
- Run 1000 event submissions, wait, repeat for 10 runs
- Realistic with respect to GEMINI, artificial conditions for real workflow application

Latency - Delayed event submission

Delayed event submission test (code example)

```
for (int r = 0; r < runs+1; r++) {  
    time = System.currentTimeMillis();  
    for (int i = 0; i < iterations; i++) {  
        WorkflowSensor.event("Test run: "+(i+1)+ " START");  
        Thread.sleep(500);  
        WorkflowSensor.event("Test run: "+(i+1)+ " CODEREGION1");  
        Thread.sleep(500);  
        WorkflowSensor.event("Test run: "+(i+1)+ " CODEREGION2");  
        Thread.sleep(500);  
        WorkflowSensor.event("Test run: "+(i+1)+ " FINISH");  
    }  
    results[0][r] = (System.currentTimeMillis() - time)/1000f;  
    //run a control run without event submissions for overhead measurement  
}
```

Latency - Delayed event submission

A control run is used to obtain a measurement of the overhead, without taking into account any submissions. This makes it possible to calculate time per event. Implemented the same as previous code without event submissions.

Latency - Delayed event submission

Results for 5 runs, 50 submissions (4 events) per run:

Run #	Real Run	Control Run	Difference
	(sec)	(sec)	(sec)
1	78.934	75.150	3.784
2	78.786	75.152	3.634
3	78.680	75.152	3.528
4	78.716	75.152	3.564
5	78.776	75.149	3.627
Average	78.778	75.151	3.627
Std Dev	0.097	0.001	0.098

Time/event (msec)

18

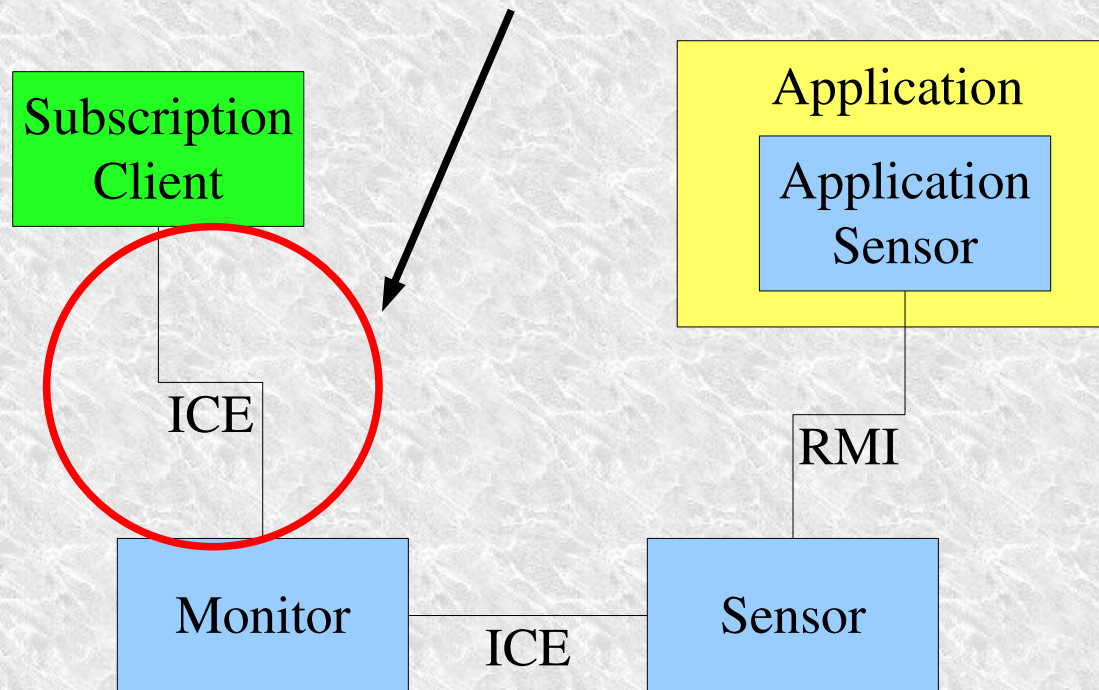
Latency - Event subscription

Event subscription:

- Create an instrumented application that submits 1000 events without any other code/overhead
- Create a monitor subscriber to receive and time events
- Insert timing statements in subscription client to measure time to receive each event
- Run 1000 event submissions, wait, repeat for 10 runs

Latency - Event subscription

Evaluation of ICE
communication channel
between client and sensor



Latency - Event subscription

Importance of overhead measurement:

- Code executed in receive() statement of monitor subscriber is used to display/store data only
- Additional overhead was added with timing statements and counters
- Runtime for entire receive() statement was measured each time it was called to see the effects of overhead

Latency - Event subscription

Event subscription test (code example)

```
public void receive(String id, String data) {  
    overhead = System.currentTimeMillis();  
    if (iterationsLeft == 1) {  
        //finish timing and store results for run  
    } else if (iterationsLeft == 0) {  
        //begin timing and reset count variable  
    }  
    iterationsLeft--;  
    totOverhead += (System.currentTimeMillis()-overhead);  
}
```

note: some trivial code excluded to simplify explanation (storing/display etc).

Latency - Event subscription

Results for 10 runs, 1000 submissions per run:

Run	Time (sec)	Overhead (sec)
1	31.108	0.007
2	30.114	0.008
3	30.072	0.002
4	27.904	0.006
5	25.283	0.003
6	23.762	0.003
7	22.434	0.009
8	23.184	0.004
9	23.181	0.009
10	21.529	0.008
Average	25.857	0.006
Std Dev	3.607	0.003

Time/Event

26

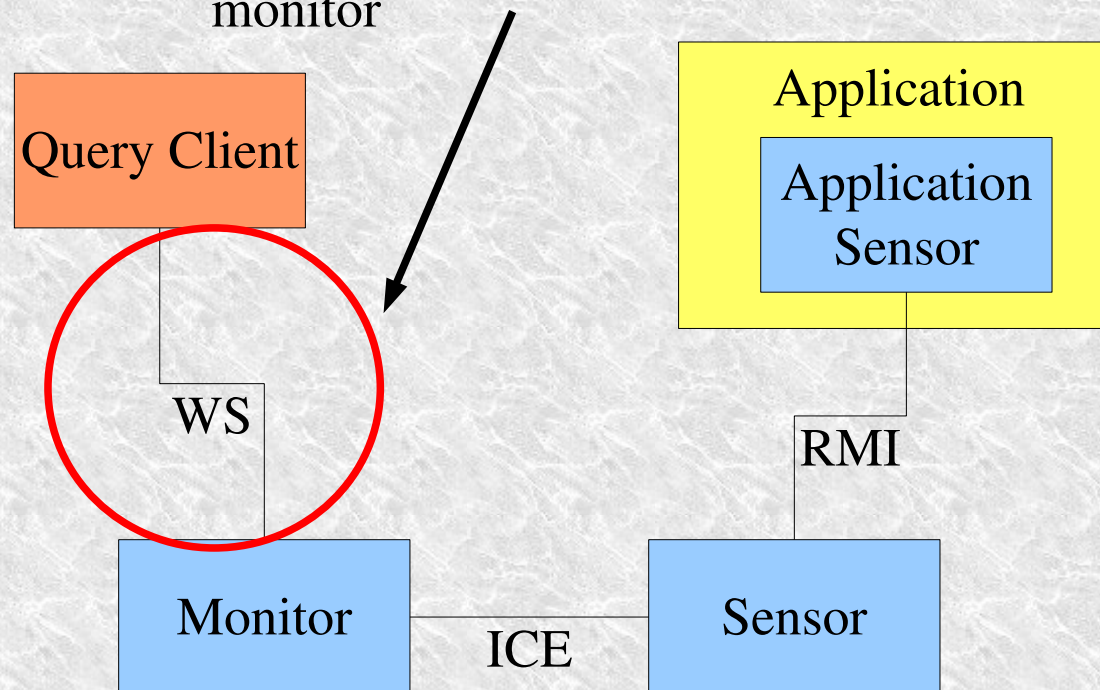
Latency – Query response

Query response:

- Create an instrumented application that submits 1000 events without any other code/overhead to a test sensor
- Use GEMINI-query method to retrieve 1000 events (stored by monitor)
- Insert timing statements to measure time to send 1000 events
- Repeat for 10 runs

Latency - Event subscription

Evaluation of Web Service
communication channel
between client and
monitor



Latency - Query response

Query response test (code example)

```
for(int d=0;d<10;d++) {  
    time = System.currentTimeMillis();  
    QueryTest client = new QueryTest(contact);  
    //set up data reception  
    while ((line = bfr.readLine()) != null) {  
        statement.append(line);  
    }  
    String [] data = client.queryData(statement.toString());  
    finalTime = (System.currentTimeMillis()-time)/1000f;  
    total += finalTime;  
}
```

note: some trivial code excluded to simplify explanation

Latency – Query response

Results for 10 runs, 1000 events sent for one query:

Query	Query Time (msec)
1	4549
2	173
3	223
4	387
5	203
6	151
7	219
8	140
9	155
10	270
Average	647
Std Dev	1373
Average (no Query 1)	213
Std Dev (no Query 1)	77

Latency/Throughput – Future testing

Modification to sustained event submission:

- Create instrumented workflow application to send events or execute arbitrary code with a given frequency
- Add a ratio or percentage variable to indicate at what frequency real events are sent (using a random number generator)
- Analyze results with respect to set ratio
- This will allow us to create various scenarios with varying density of submitted events

Scalability – Future testing

Possible scenarios:

- Multiple clients
 - Evaluate GEMINI's ability to respond to many queries and subscriptions to growing number of clients.
 - Multiple clients could be installed across a number of hosts to simultaneously query or subscribe to monitors/sensors for information.
- Multiple sensors
 - Evaluate GEMINI's ability to deliver information to client/s from growing number of sensors.
 - Multiple sensors, installed on different hosts to send increasing amounts of data through monitor/sensor structure.

The End!

Questions? Comments?