

A Presentation of our Work

The Swedish Interns

2016-08-18

Created for NVI Inc. at the Goddard Space Flight Centre

Table of Contents

Weeks 1-2: Learning Fortran

Weeks 3-7: Our First Project

Week 3-4: Testing I/O performance

Weeks 5-7: Implementation

Results for Project One

Weeks 7-9: Our Second Project

Result for Project Two

Week 10: Returning Home

Weeks 1-2: Learning Fortran

Calculator

```
erik@Antergos-Laptop ~/Programming/git/GSFC_Internship/server_client >./calculator_client.out
Please enter the same port number as for the server (e.g. 55555).
55555
Usage: Write two numbers followed by an operator (add, sub, mul, div).
Only integer answers are supported. (2 4 div will return 0)
1238 4883 add
The answer is:          6121
erik@Antergos-Laptop ~/Programming/git/GSFC_Internship/server_client >./calculator_client.out
Please enter the same port number as for the server (e.g. 55555).
55555
Usage: Write two numbers followed by an operator (add, sub, mul, div).
Only integer answers are supported. (2 4 div will return 0)
128 412 mul
The answer is:          52736
erik@Antergos-Laptop ~/Programming/git/GSFC_Internship/server_client >./calculator_client.out
Please enter the same port number as for the server (e.g. 55555).
55555
Usage: Write two numbers followed by an operator (add, sub, mul, div).
Only integer answers are supported. (2 4 div will return 0)
121 11 div
The answer is:           11
erik@Antergos-Laptop ~/Programming/git/GSFC_Internship/server_client >./calculator_client.out
Please enter the same port number as for the server (e.g. 55555).
55555
Usage: Write two numbers followed by an operator (add, sub, mul, div).
Only integer answers are supported. (2 4 div will return 0)
2138 1312 sub
The answer is:           826
erik@Antergos-Laptop ~/Programming/git/GSFC_Internship/server_client >_
```

Figure 1: Example usage of the TCP calculator.

Weeks 3-7: Our First Project

Problem Description

Rewrite how `globl/solve` handles its passing of data to and from `usrpartials` and `usrprogs`.

By minimizing disc I/O we want to increase the speed at which data is sent.

Week 3-4: Testing I/O performance

- A couple of contenders:

- A couple of contenders:
 - Read/Write with files

- A couple of contenders:
 - Read/Write with files
 - Read/Write with pipes

- A couple of contenders:
 - Read/Write with files
 - Read/Write with pipes
 - Sending/Receiving with TCP Sockets

- A couple of contenders:
 - Read/Write with files
 - Read/Write with pipes
 - Sending/Receiving with TCP Sockets
 - Sending/Receiving with OpenMPI

- A couple of contenders:
 - Read/Write with files
 - Read/Write with pipes
 - Sending/Receiving with TCP Sockets
 - Sending/Receiving with OpenMPI
 - Sending/Receiving with ZeroMQ

1. The producer generates a list of length n and fills it with integers.

Performance Test

1. The producer generates a list of length n and fills it with integers.
2. The producer writes the list to file (or sends it over the designated transfer protocol).

Performance Test

1. The producer generates a list of length n and fills it with integers.
2. The producer writes the list to file (or sends it over the designated transfer protocol).
3. The consumer reads (or receives) the list.

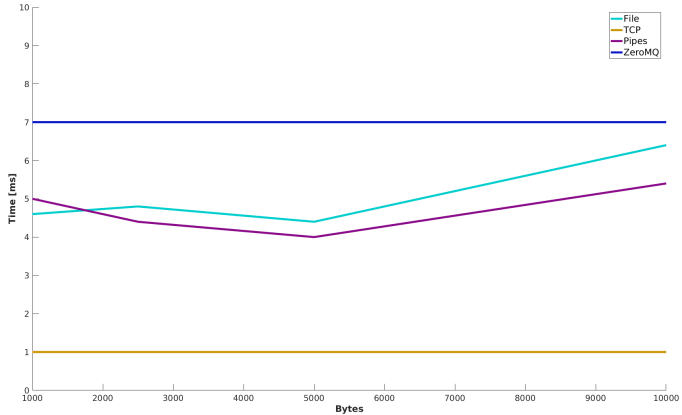
Performance Test

1. The producer generates a list of length n and fills it with integers.
2. The producer writes the list to file (or sends it over the designated transfer protocol).
3. The consumer reads (or receives) the list.
4. The consumer squares each int in the list and sends it back to the producer.

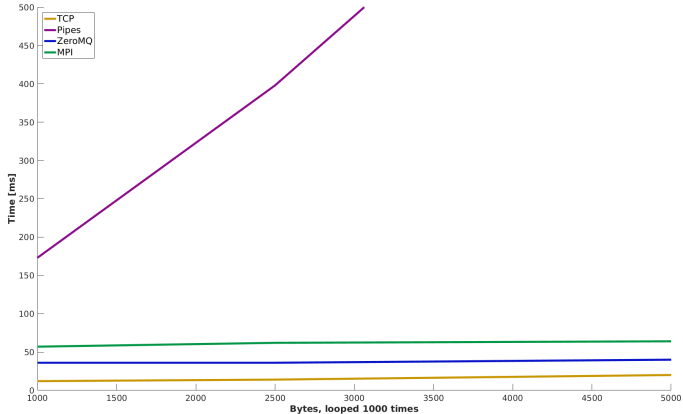
Performance Test

1. The producer generates a list of length n and fills it with integers.
2. The producer writes the list to file (or sends it over the designated transfer protocol).
3. The consumer reads (or receives) the list.
4. The consumer squares each int in the list and sends it back to the producer.
5. The producer reads (or receives) the modified list.

Result for I/O Performance



Result for I/O Performance



Result for I/O Performance

- TCP was the fastest, but the most difficult to implement.

Result for I/O Performance

- TCP was the fastest, but the most difficult to implement.
- Since we assumed a lot of data would be passed we opted for ZeroMQ due to its presumptive ease of use and performance.

Weeks 5-7: Implementation

- Installation of Software on **bootes**

- Installation of Software on **bootes**
- Porting our code to ifort

Implementation

- Installation of Software on **bootes**
- Porting our code to ifort
- A lot of coding

Results for Project One

Results for Project One

Running userpartials for 200 observations.

Old version of globl/solve: 8 minutes 47 seconds

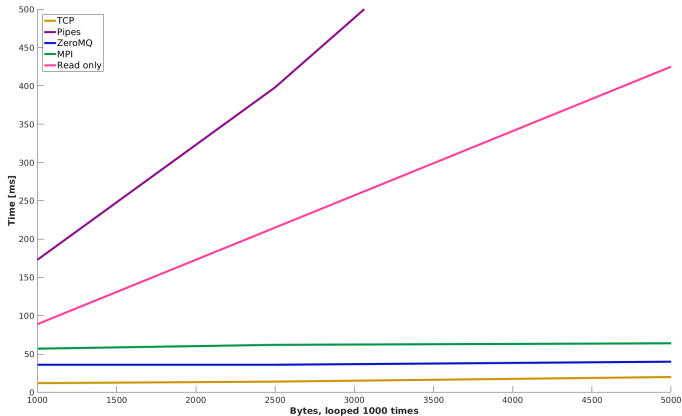
New version of globl/solve: 9 minutes 5 seconds

Results for Project One

Why?

Results for Project One

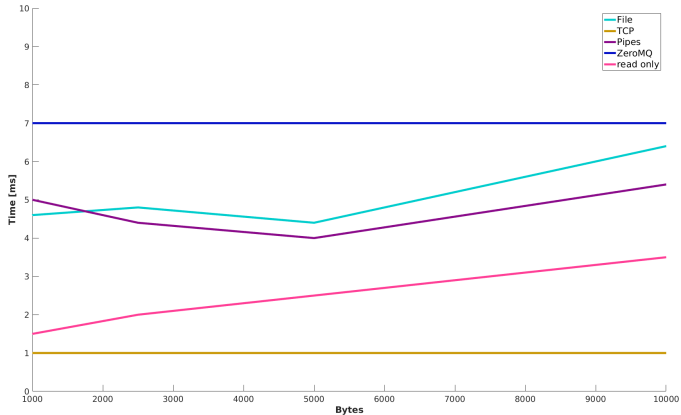
Globl/solve does not read and write a lot, it reads a lot,



but read only is still slower...

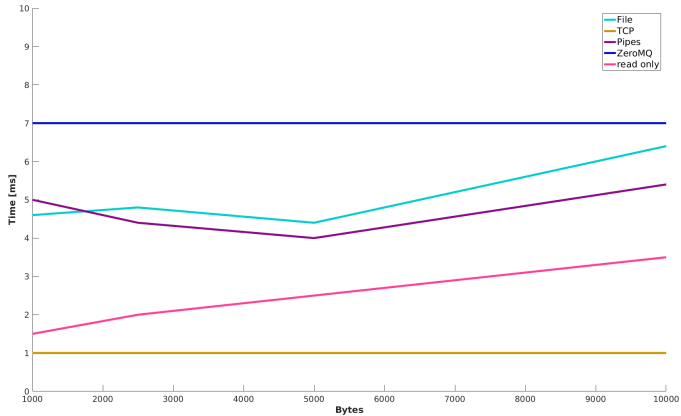
Results for Project One

Globl/solve does not read as much as we thought,



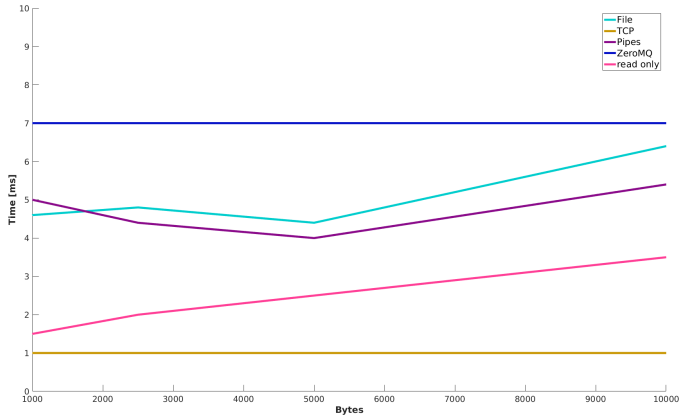
Future studies

- Remove the writes that are no longer needed.



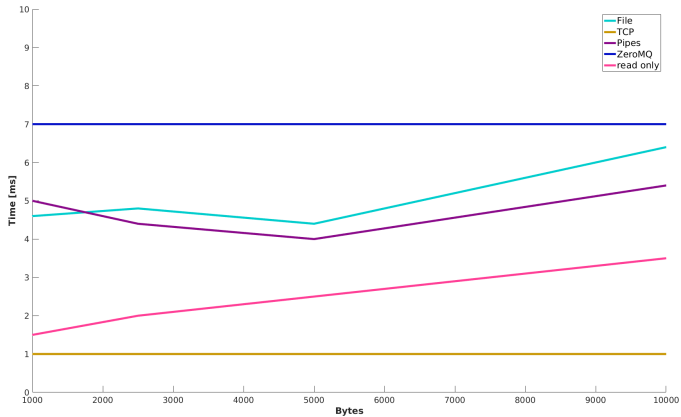
Future studies

- Remove the writes that are no longer needed.
- TCP should have been the way to go?



Future studies

- Remove the writes that are no longer needed.
- TCP should have been the way to go?
- Posix Shared Memory? Conversion between C and Fortran was non trivial.



Weeks 7-9: Our Second Project

Problem Description

The model used for determining when an antenna acquires a source is a little bit off. Sked will say that an antenna should be on target at time T but the logfiles tells us that the antenna actually was on source at $T \pm c$.

Extract the Data

Observation listing from file ./r1747.skd for experiment R1747---

Source	Start	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL	AZ	EL
name	yyddd-hhmmss	Ft	Hh	Hb	Is	Ke	Ny	On	Kv	Sh	Ts	Wz	Yg														
0834-201	16186-170000	211	71	257	20																						
3C418	16186-170000					341	74	360	24	410	45	398	29														
2052-474	16186-170332					239	80	543	6	536	57																
0808+019	16186-170341	287	72																								
1717+178	16186-170709			55	23			628	34			468	22														
1923+210	16186-170918					325	18	607	61	329	49																
0955+476	16186-171036	366	38									593	55	627	63												
2008-159	16186-171235																										
0059+581	16186-171315																										
1908-201	16186-171450					290	49	576	25	253	65																
2059+034	16186-171504																										
1144+402	16186-171552	392	37	334	17							561	50	580	69												
1759-396	16186-171642									228	44																

```
2016.186.17:01:31.00:source=2052-474,205616.36,-471447.6,2000.0,neutral
2016.186.17:01:32.00#flagr#flagr/antenna,new-source
2016.186.17:01:32.00:sx8m11a=1
2016.186.17:01:32.00:!2016.186.17:03:22
2016.186.17:01:52.00#trakl#Source acquired
```

- Correlate the log data with the Sked data.

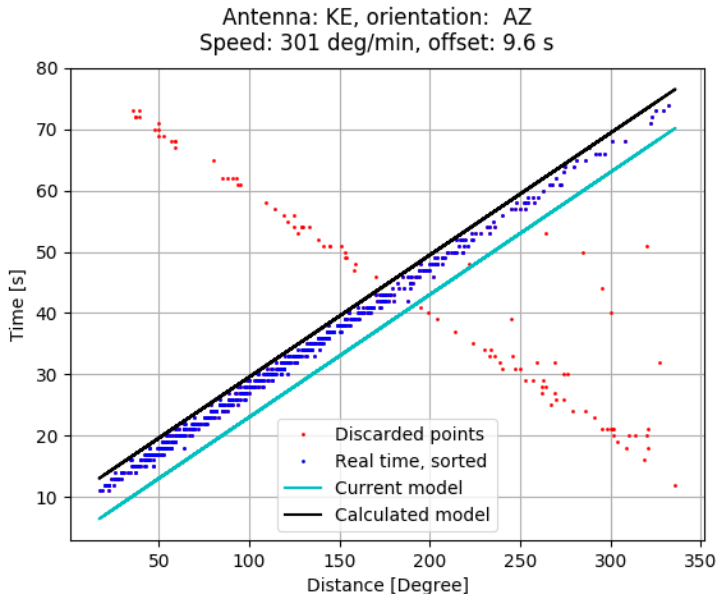
- Correlate the log data with the Sked data.
- Least Square Fit

Parse the Data

- Correlate the log data with the Sked data.
- Least Square Fit
 - Iteratively re calculate to get rid of noise.

Result for Project Two

Result for Project Two



- Fix the bug in Sked.

- Fix the bug in Sked.
- Set a more intelligent threshold.

Week 10: Returning Home

Returning Home

During our time here we have:

Returning Home

During our time here we have:

- been exposed to how it is to work at NASA and NVI.

During our time here we have:

- been exposed to how it is to work at NASA and NVI.
- learned the importance of conducting thorough research.

During our time here we have:

- been exposed to how it is to work at NASA and NVI.
- learned the importance of conducting thorough research.
- realized that patience really is a virtue...

Thank you!