The Challenge of Data Transfer

Ciaron Linstead

Potsdam Institute for Climate Impact Research

Mistra-SWECIA Seminar, 28th May 2008

linstead@pik-potsdam.de

Why data transfer?

- Programs are most flexible when they can be combined to produce useful effects
  - in Unix, "ls" lists files in a directory
  - "wc" counts word in a list
  - `ls | wc`
- the data is the interface between programs
  - output from one program is input to another
  - an idiomatic way to do things in the tradition of Unix operating system development

The problems with data transfer

- Technical
  - the programming language (C, Fortran, Python, Java...)
  - the OS (Linux, IBM AIX, Windows, Mac OS X, Sun Solaris...)
  - the architecture (Intel x86, PowerPC, SPARC)
  - communication protocols
  - programming style
- Semantic
  - does the output of one program make sense as the input to another?

Typical technical problems

- Endianness
  - the order of bytes in a word in memory:
    - Big-endian (Motorola 68000 series, PowerPC, SPARC)
      0x0A → 0x0B → 0x0C → 0x0D
    - Little-endian (Intel x86)
      0x0D → 0x0C → 0x0B → 0x0A
- Sizes of datatypes, with my compiler and PC:

```
struct test {
    unsigned char  field1;
    unsigned short field2;
    unsigned long  field3;
} __attribute__((__packed__));
```

  - 8 bytes unpacked, or 7 bytes packed
- C and Fortran arrays: row vs. column first

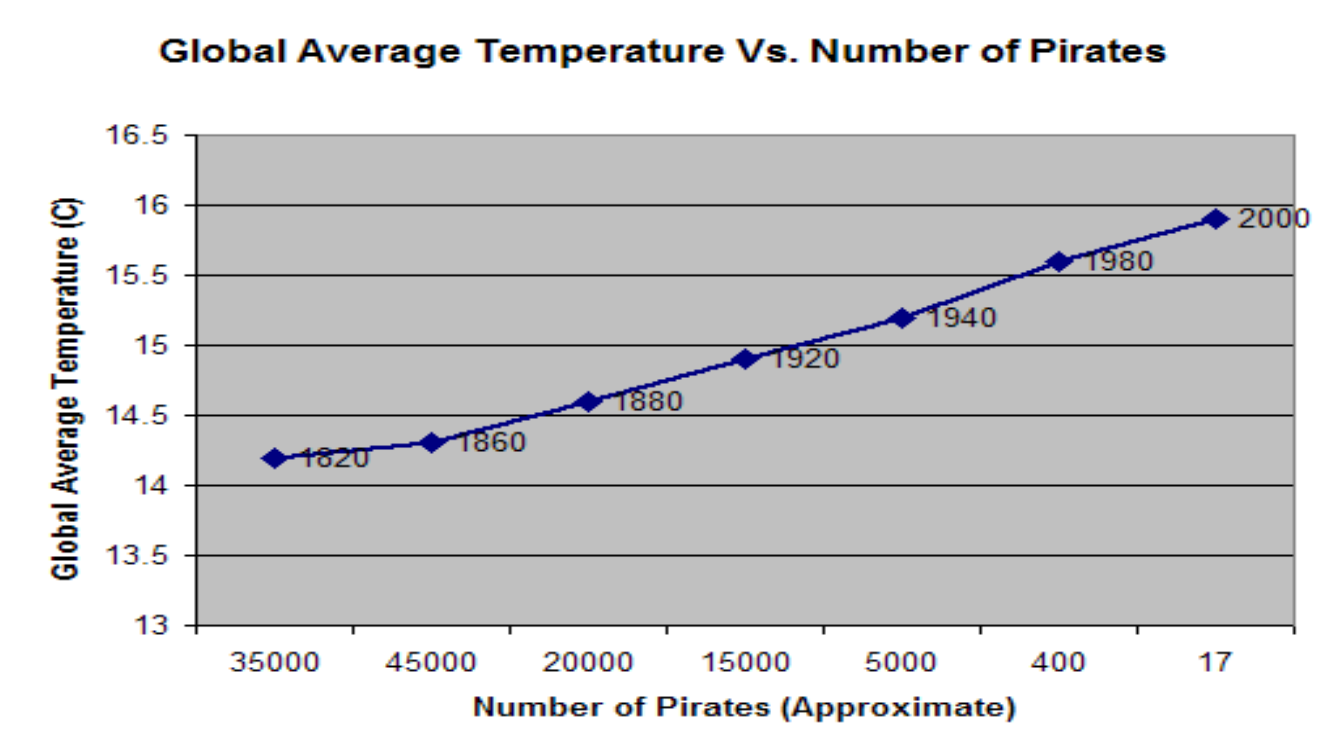
Typical technical problems

There are 10 types of people in the world...

those who understand binary and those who don't.
those who understand ternary, those who don't, and those who mistake it for binary.

Problems of model semantics

- Are units compatible?
- Does coupling make sense?



- Higher-level problems, but maybe we can help

So, we need to think about...

- the work needed at each end...
  - establish point-to-point connections
    - sockets, files, shared memory?
    - the same protocol at each end?
  - check sizes of data
  - share information about layout of data in memory
  - normalise data
- ...but this is hard
  - it's complicated; more bugs are introduced
  - it's repetitive and boring

An aside on software design

- A software design philosophy
  - software development is about managing complexity
  - do one thing, do it well
  - simple design patterns are the most successful
    - Rule of Thumb: an API should have about 7 functions
  - design and build software to be tried early
  - bottom-up development

The Art of UNIX Programming*

- Rule of Modularity
  - simple parts connected by clean interfaces
- Rule of Composition
  - design programs to be connected to other programs
- Rule of Representation
  - Fold knowledge into data, so program logic can be stupid and robust
- Rule of Extensibility
  - Design for the future, it will be here sooner than you think

* Eric Raymond, 2004

Solving the data transfer problem

- Technology exists for doing this, and often much more, for example
  - OASIS (Ocean-Atmosphere-Sea Ice-Soil)
  - CORBA (Common Object Request Broker Architecture)
  - MCT (Model Coupling Toolkit)

But...

- OASIS:
  - Ocean-Atmosphere-Sea Ice-Soil
- CORBA: Common Object Request Broker Architecture
  - a standard for connecting software components defined by the Object Management Group (OMG)
  - The specification document *index* is 14 pages and 750kB
- MCT:
  - API (Application Programming Interface) document is 284 pages

Typed Data Transfer (TDT) Library

- software library for transferring data with known type (e.g. integer, floating point, or structures)
- programs use simple statements (like "read" and "write") to move data around
- data should have a description
- connections should have a description
- make all this transparent and easy to extend

TDT - data descriptions

- sending side

```
int mymatrix[3][4]={0, 1, 2, 3,
                    4, 5, 6, 7,
                    8, 9, 10, 11};

write(socket, mymatrix, size_of(mymatrix));
```

- receiving side:

```
{0,  1,          {0, 1, 2, 3,  4,  5,
 2,  3,           6, 7, 8, 9, 10, 11}
 4,  5,
 6,  7,
 8,  9,                ???
10, 11}
```

- How do we interpret "mymatrix"?

TDT - data descriptions

- TDT knows about data types:
- ```
  <decl name="mymatrix">
        <array size=3>
              <array size=4>
                    int
              </array>
        </array>
  </decl>
  ```
- `tdt_write(mymatrix, "mymatrix", connection);`
- Side effect: we're writing down our interface
  ```

TDT - data descriptions

- Logic is separate from data
- This works transparently where row/column order are different
  - e.g. in C, matrices are organised Row x Column in Fortran, they're Column x Row
- Also handles the endian problem

TDT – connection descriptions

- creating socket connections by hand...

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(char *msg)
{
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[256];
    if (argc < 3) {
       fprintf(stderr,"usage %s hostname port\n", argv[0]);
       exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
         (char *)&serv_addr.sin_addr.s_addr,
         server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,&serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR connecting");
```

TDT – connection descriptions

- Or...
  - `tdt_open(connection);`
- Connections, like data, are described externally:

```
<channel name="clnt_to_serv”>
    mode="out"
    host="pc61.pik-potsdam.de"
    port="2424"
    type="socket"
    datadesc="datadesc.xml">
</channel>
```

TDT – connection descriptions

- We can also use intermediate files for communication
  - dump data for use later
  - connect programs that can only read files, e.g. GAMS (General Algebraic Modeling System) code
- And we don't have to re-write chunks of our model to do so
  - `type=socket` or `type=file`

TDT in service

- @PIK: prototyping the Climber 3 alpha climate system model
- @PIK: modularisation of Integrated Assessment models
- @PIK: Prototype internet-coupled models with Centre for Novel Computing, Manchester
- Netherlands Environmental Assessment Agency hydrological modelling
- Geoforschungs-Zentrum Potsdam hydrological modelling

TDT in service

- Distributed Model Coupling with Centre for Novel Computing, Manchester
  - communication over the internet via Secure Shell (ssh)
  - TDT-based models at PIK
  - CNC's BFG (Bespoke Framework Generator)-based models at CNC
  - Minimal changes to connection descriptions to get this working in a distributed way
    - *and no code changes!*

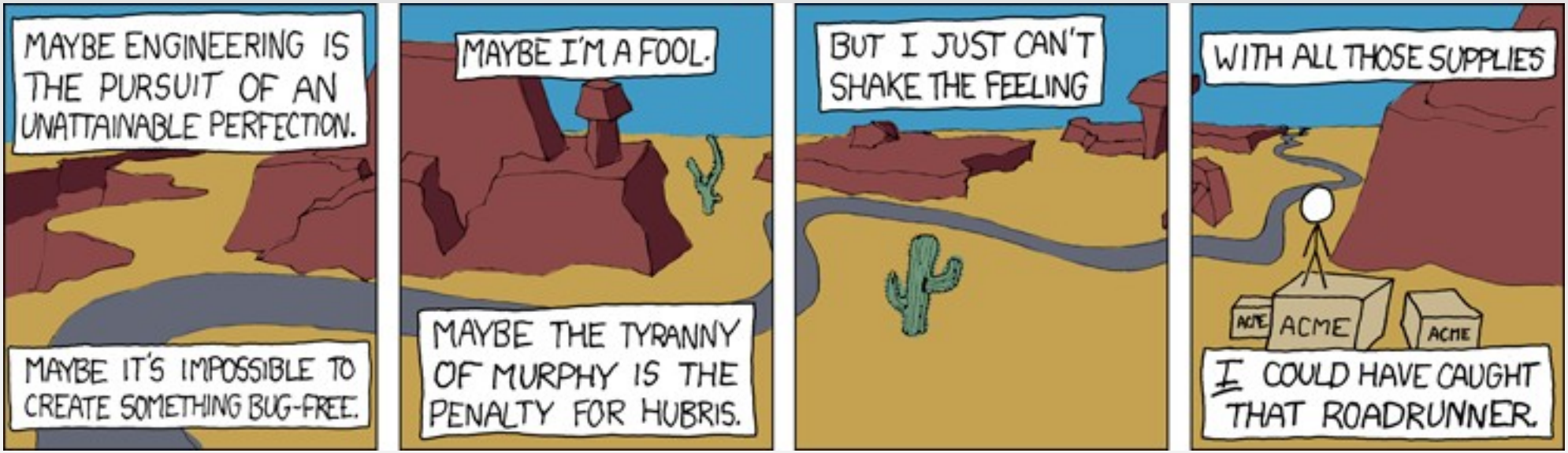
TDT in service

- 3000 lines of C code
- 7 API functions (configure, open, read, write, close, plus a couple of specialised functions)
- Interfaces exist for Fortran, Python, Java, MATLAB, Visual Basic
  - but any language supporting foreign function interfaces can be added
  - ...and some without (e.g. GAMS)
- TDT is Free Software (GNU GPL)

Conclusions

- Design software to be
  - easy to learn
  - easy to use
  - easy to extend
- Use TDT to
  - rapidly prototype a coupled model system, across different computer platforms and networks
  - abstract away low-level, error-prone technical work
- No Silver Bullet

Thanks!


MAYBE ENGINEERING IS THE PURSUIT OF AN UNATTAINABLE PERFECTION.
MAYBE IT'S IMPOSSIBLE TO CREATE SOMETHING BUG-FREE.
MAYBE I'M A FOOL.
MAYBE THE TYRANNY OF MURPHY IS THE PENALTY FOR HUBRIS.
BUT I JUST CAN'T SHAKE THE FEELING
WITH ALL THOSE SUPPLIES
ACME
ACME
ACME
I COULD HAVE CAUGHT THAT ROADRUNNER.

References

- These slides, the TDT software (including sample code) and user guide: http://www.pik-potsdam.de/software/tdt/
- OASIS:http://www.cerfacs.fr/globc/software/oasis/
- CORBA: http://www.corba.org
- MCT: http://www.mcs.anl.gov/mct
- The Art of UNIX Programming, Eric S. Raymond, Addison-Wesley 2004
- GNU GPL: http://www.gnu.org/licenses/gpl.html
- http://xkcd.com for the comics
- No Silver Bullet - Essence and Accidents of Software Engineering, Fred Brooks, 1986