The True Bottleneck of Modern Scientific Computing

in astronomy… and not only

Igor Chilingarian (CDS / SAI MSU)
Ivan Zolotukhin (ObsPM / SAI MSU)
A few facts about us

• Ivan
  – “fresh” postdoc in astrophysics (PhD+1)
  – enterprise-level web developer, team manager, DB expert

• Igor
  – advanced postdoc in astrophysics (PhD+4)
  – hacker: any OS/platform, from Assembler to IDL, from microcontrollers to mainframes

• Both
  – participated in open source software development related to PostgreSQL
Data & CPUs vs. Researchers

• Astronomical data growth in time is an exponential avalanche

• Computational resources and data storage also grow exponentially (Moore’s law)

• However, we do not see an exponential growth of research results in astronomy

WHY? Where is the bottleneck?
Code written by astronomers

Often it is:

• Fortran95, 90, 77 (or even 66… or even 4)
• “GOTO” once every 10-20 lines
• Undocumented, full of “intuitive” algorithmic solutions sometimes being quite far from what CS students learn
• Multi-layered code “structure”
  – layers written during different periods of time serving as different interfaces to do the same

• But at the end the code does what it is supposed to (according to its author)
Code written by [good] engineers

Usually it is:

• “Real” language: C/C++/Java

• No “intuitive” algorithmic solutions because of the author who at least heard about Donald Knuth’s books

• Well organized, easy to read

• Documentation strongly depends on the project manager – from none to perfect

• But the author does not understand physical principles (instrumentation) behind the code – *possible surprises*
A real nightmare: databases

[Small] DBs developed by astronomers

• Custom implementation of re-invented indexing solutions and primitive requests to the data. Yep, in Fortran… or IDL!

• If existing DBMS solutions are used, then the “database” contains just one or several flat tables without mutual links.
  – No consistency checks via DB constraints

• Terrible user interfaces, both API & web
Some funny examples (1)

An unnamed galaxy catalogue

- No access interface on the web
- Distributed as a set of dozens of FITS tables with a total volume >10Gb
- IDL access routines for these tables
  - Huge requirements for memory if one wants to use the whole catalogue at once
  - Very slow...
- No SQL access - why?
  - “Astronomy stops where SQL starts”, PI says
Some funny examples (2)

An unnamed astronomical database

- DB admin interface in Fortran77 had a function with **264** arguments
- How to delete a record from a table?

  DELETE FROM tab1 WHERE fld1=‘val1’?

  NO! Too trivial and flexible, everyone can do!

  ```
  pg_dump -t tab1 mydb | grep -v val1 | \\
  pg_restore -c mydb
  ```

- Record of the development pace: 4 (!) stable (!) releases in one day!
Some funny examples (3)

An unnamed astronomical database

- The most striking example, a database connecting to itself from within a stored procedure

ISN'T THAT... FAIL?!!?
Some positive examples (1)

Technologically advanced resources with intuitive user interfaces

• HLA – Hubble Legacy Archive
  – VO interfaces as a hidden middleware
  – XSLT transformation of VOTables
  – Advanced visualisation

• SDSS CasJobs

• GalexView

• Millennium Simulation database

• GalMer – galaxy mergers database
Some positive examples (2)

**Computational packages**

- **GADGET-2** (*V. Springel*)
  - extensible and relatively well documented

- **SExtractor** (*E. Bertin*)
  - intuitive configuration but outdated docs
  - relatively clearly written code

**Data manipulation / visualisation**

- **TOPCAT / STILTS** (*M. Taylor*)
- **ALADIN** (CDS, *mostly P. Fernique*)
- **DS9** (*W. Joye*)
What can we conclude?

- All examples mentioned in the last two slides were developed
  - Either by professional astronomers with very strong IT/CS background
  - Or by IT/CS professionals working in a close link with astronomers and understanding astronomy
- One cannot simply hire an industrial software developer and make him/her developing astronomical software and/or an archive and/or a database
Solution?

• Change the teaching process for students in astronomy/astrophysics

• Make advanced courses in algorithms, programming, software development and maintenance mandatory in the education of astronomers/physicists
  – Fortran is now obsolete and we have to accept this. Instead of teaching Fortran programming, teach interfacing of Fortran routines in C/C++
Our exciting future

The moment, when the CPU power and data storage and transfer become the bottleneck of scientific computing will become the beginning of a new “exponential” scientific era and we will have an avalanche of discoveries.