Towards secure mobile computation for (astronomical) data centres

James Cheney (with Bob Mann)

University of Edinburgh
The problem

- Historically, scientists have managed data in ad hoc ways.
- In astronomy (and many other disciplines), databases are now necessary for managing scientific datasets.
- Conversely, database technology is enabling new science.
- Moreover, the way science is performed is also changing.
The problem

- Astronomical sky surveys (such as WFCAM) are under way
- Already producing around 10s-100s of TBs of data per year
- Individual astronomers do not have the resources to manage such datasets locally.
- Instead, they are to be managed by astronomical data centres (e.g. ROE)
- Similar situations in other sciences (e.g. bioinformatics)
- How to provide individual scientists with the ability to analyze this data?
Traditional model

- Traditional research model:
  - Scientist visits observatory
  - Scientist makes/records observations (in ad hoc/text formats)
  - Scientist takes data back to home institution
  - Scientist does analysis locally
  - Scientist publishes a paper
  - Scientist puts data in a box under desk.

- Data moves to code
New model

New research model:
- Observatory makes a large number of observations on behalf of community
- Observatory stores results in DBMS
- Scientist analyzes the data
- Scientist publishes paper
- Scientist makes the results of the analyses available to other scientists

Code moves to data
Security issues

- Providing remote shared access to any resource on a network introduces many risks.
  - Loss or (silent) modification of irreplaceable observations
  - Misuse for illegal or non-research purposes
  - Theft or damage of IP (plagiarism, sabotage, accident)
  - Crashes, denial of service, resource hogging
Security issues

- Standard security techniques suffice for many of the risks.
- For example, non-networked archival backups help ensure the integrity of irreplaceable data.
- Grid-based credential systems being developed to manage authentication.
- Access control systems help protect users of shared resources from each other.
- Existing Grid security infrastructure is in place/in development for many of these issues.
- However, security issues relating to mobile code (moving code to data) are still an active research area.
Key requirement

- The ability to run data analyses on remote data center machines is a key requirement.
- We want to provide users flexibility/efficiency comparable to having *their own copy* of the database while minimizing expenses related to administrative overhead and security risks.
- These capabilities go *far beyond* the ability to query the data.
First-class access to data

- If users had their own copy of the DB, they could:
  - Write customized applications that interact with the database
  - Install *stored procedures* for moving processing inside the database
  - Create new tables for temporary results or to share the results of analyses with others
  - Develop customized indexing schemes for the data

- Providing all these abilities to legitimate users while maintaining an acceptable level of risk and management overhead is a significant challenge.
Mobile code is a huge security risk

- Gaining the ability to run code on a remote machine is a big step towards gaining control of it.
- So, it would be very risky to give unknown users the ability to run arbitrary code.
- Even authorized users may accidentally submit programs that hog resources or crash the server.
- Or authorized users’ accounts could be compromised.
- Even though the DB is archived, restoring from backups can take hours, denying service to legitimate users.
- (Imagine this happening right before a paper deadline!)
What can we do now?

- Obviously, safely sharing access to networked computers is a well-studied problem.
- What can we do now?
- Is it enough?
Scientist moves to data

- No remote access: require users to visit the data centre personally
- Expensive! Defeats purpose of networking
- But minimal risk.
Data moves to code

- Require users to download data (or send out tapes)
- Still expensive! Users have to maintain DB locally, which requires expensive expertise.
- Alternatively, maintain multiple data centres, so that each user can visit a local one
- Also too expensive.
- But risks still low.
Query interface

- Allow users to send DB queries, download results, run further analysis locally
- Several systems already allow this.
- Inexpensive, low risk.
- But provides limited analysis capability.

![Diagram of query interface]

Secure mobile computation – p.14/2
User accounts

- Give each user an account at the data centre; use traditional OS security or “virtualization”
- Users log in and develop/run programs locally
- Less expensive, but administrator still needs to manage accounts, monitor server, kill runaway processes/queries
Reference monitoring

- Idea: Wrap DB server with a *reference monitor* that provides illusion of full DB access: “MyDB” (Szalay, Gray, et al.)

- Users can create local tables, add stored procedures for their own use

- Amounts to re-implementing OS access control system inside DB
None of the approaches comes close to providing “first-class” access to the database to legitimate users for a reasonable cost.

The first two approaches provide full access, but very expensively

the third provides limited access cheaply

the fourth provides greater (but still limited) access but still expensive

the fifth is reasonable, but still under development

Conclusion: Still some work to do.
Recent research directions

- Mobile code security has received a lot of attention (e.g., Java, C#)

- *Reference monitoring* is a common approach to security

  Idea: All accesses to resource are controlled by system layer (e.g. OS, DBMS)

  which tracks uses and decides whether to allow access

- Recent improvement: *inlined reference monitoring*, which rewrites mobile code so that it *cannot* violate the security policy
An idea

- Users submit programs to the DB server
- Server uses IRM to rewrite each program so that it is guaranteed to obey security policy
- Advantage: Unlike MyDB, don’t need to modify underlying DBMS
- Disadvantage: Dynamic checks slow down programs
Recent research directions

Proof-carrying code: another recent approach

Idea: Code producer must **certify** (formally prove) that code has some **safety property**

such as “only accesses allocated memory” (= does not crash) or memory/processing limits.

Code consumer can **check** the proof before running the code
An idea

- Idea: Use PCC to certify “database programs”, which run inside DB as *stored procedures*.
- Power users write & submit procedures, e.g. custom indexing schemes
- Ordinary users can use the procedures in plain SQL queries.
- Stored procedures run at full speed.
MRG, Mobius, and Request

- We have a lot of local expertise on PCC!
- MRG (Mobile Resource Guarantees) is a recently completed EPSRC funded project at UoE.
- Developed extensions to proof-carrying code for certifying *resource usage*.
- Mobius and Request are recently-begun projects to implement PCC for “full” Java and apply MRG research to eScience.
- Astronomical (and other scientific) data analysis problems may be a “good fit” for PCC/MRG secure mobile code techniques.
Gotchas

- Work in progress!
- Little work on language-based security for databases
- PCC research has focused on simple, easy-to-analyze functional programming languages (that nobody else uses)
- Scaling up to full Java is still a hard research problem
- Forget about C/C++ or proprietary languages (IDL, Matlab).
- Question: Are scientists willing to learn/use slower or higher-level languages for their analyses?
Gotchas

- Work in progress!
- Little work on mobile code security for databases
- PCC research has focused on simple, easy-to-analyze functional programming languages (that nobody else uses)
- Scaling up to full Java is still a hard research problem
- Forget about C/C++ or proprietary languages (IDL, Matlab).
- Question: Are scientists willing to learn/use slower or higher-level languages for their analyses?
  - Even if it’s the only (secure) way to allow access...?
Beyond astronomy

- We’ve focused on astronomical data analysis
- But same problems likely to appear in any discipline with large-scale data needs

- Some examples
  - Bioinformatics (human and mouse genome)
  - Social sciences (allowing remote access to demographic data): privacy/anonymity is an issue
  - Geospatial/planetary sciences
Help!

- We need **examples** of analyses people would like to run
- or **case studies** (use cases) describing what people do now vs. would *like* to be able to do
- in order to be sure we’re doing something that is relevant to real situations
- Actual code would be especially helpful
- (or actual code with “interesting” algorithmic content removed...)
- Realistic examples needed at some point to build a working system
Conclusions

- Providing safe, first-class access to astronomical (and other scientific) databases is challenging.
- Recent work on language-based security may be applicable.
- There may be further interesting security research to do here.
- Or the current state of the art may be enough.
- To know for sure, need examples & use cases from potential users.