Security (2) and Some Takeaway (Lecture 26, cs262a)

Ali Ghodsi and Ion Stoica,
UC Berkeley
December 2, 2020

Today's Lecture (1/2)

CryptDB: Protecting Confidentiality with Encrypted Query Processing, Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan
https://people.csail.mit.edu/nickolai/papers/popa-cryptdb.pdf

Opaque: An Oblivious and Encrypted Distributed Analytics Platform, Wenting Zheng, Ankur Dave, Jethro G. Beekman, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica
https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zheng

Today's Lecture (2/2)

Query/process data running on an untrusted infrastructure

Public cloud a big part of the motivation

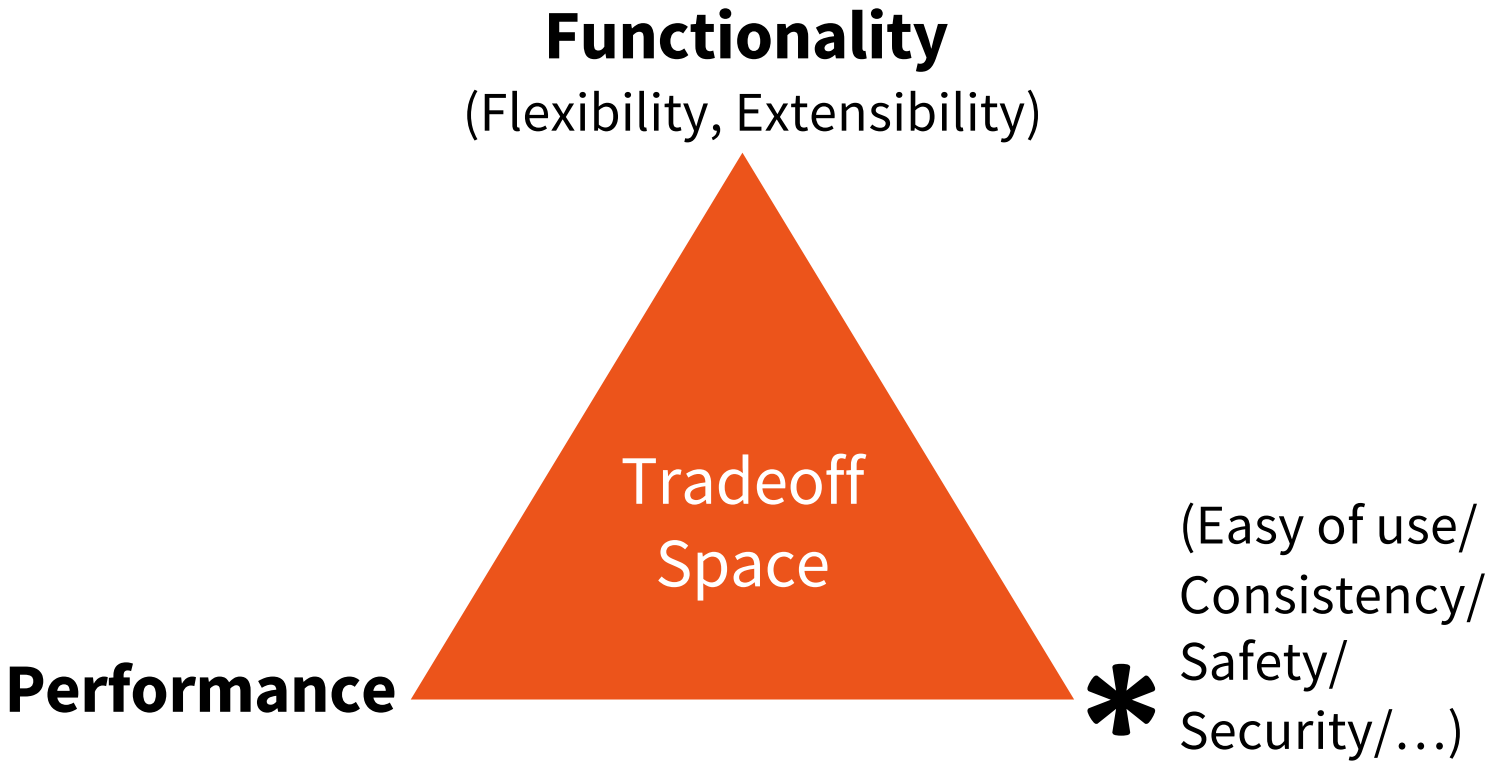
Two solutions

- Data is encrypted on server, so server cannot "see" it
- Data is protected by a hardware enclave that runs the client's code; no other code (even OS) can "see" data

Need to be worried about side channel attacks.

Designing a system

An exercise in picking the point in tradeoff space…



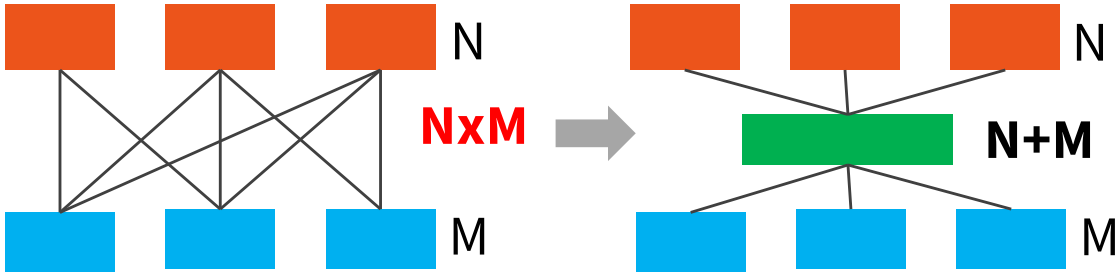
Tradeoff space examples

	Performance	**Functionality** (Flexibility / Extensibility)	* (Easy of use / Consistency/ Safety / Security/ Reliability…
SQL			Easy of use 
Unix			Easy of use / security 
Internet			Reliability / QoS 
Microkernels			Safety / Security 
MapReduce			Scalability / reliability 
Spark			Scalability / reliability 
TensorFlow or MPI			Easy of use 

Big Theme: Indirection

Add an intermediate layer to

- Simplify system design
- Decouple evolution of lower and higher layer



Examples:

- Internet protocol (IP): between Transport and Link layers
- OS: between app and hardware
- VM: between OS and hardware
- LLVM IR: between high-level language and machine code
- Logical query plan: between SQL query and physical execution
- Abstract Device Interface: between MPI app and communication infra
- Spark: between data app and cluster (abstracts away parallelism)
- …

Big Theme: End-to-end arguments

Think twice about implementing functionality at lower layer:

- Only if functionality shared by many apps
- Only if it doesn't hurt performance of apps that don't need it

Examples:

- IP (wide-area routing; shared by all transport protocols, e.g., TCP and UDP)
- Microkernel, Exokernel (IPC, protection / isolation, maybe scheduling)
- RISC processor (basic memory and arithmetic operations)
- ADI for MPI (several functions vs 100+ functions)
- …

Big Theme: Specialization

Improve one dimension without impacting others

- Idea: Leverage semantics about workloads to specialize implementations!

Examples:

- SQL: focus on querying structured data (improve performance)
- CRDT: focus on commutative operations (improve performance)
  - Also recall coordination avoidance
- GFS: focus on large, append only file systems (improve scalability)
- Idempotent operations (improve fault tolerance)
- GPU: focus on graphic, deep learning (improve performance)
- TPU: focus on deep learning (improve performance)
- …

Some advice on system design…

Typically many constraints, many goals you want to satisfy…

Resist the urge to satisfy all of them, prioritize!

Always start with the simplest design first

- Much easier to add features than remove them!
- Allow you to get early feedback, then iterate fast
- Minimalist API, clear semantics

Correctness first, optimization second

- Much easier to optimize latter than add fault tollerance!

“Make simple things simple and complex things possible” – *Alan Kay*

- Enable users to get some simple things done out of the box → adoption
- Enable experts to experiement → you’ll learn a lot

Project

Poster session: 9-11am PT, Wednesday 12/16

Project reports due: 11:59PT, Friday 12/18