Recovery-Oriented Ground Systems

James Cutler

Space Systems Development Laboratory
Software Infrastructures Group
Stanford University

GSAW 2003

Introduction

- Summary of past computer system research goals
  - Goal #1: Improve performance
  - Goal #2: Improve performance
  - Goal #3: Improve cost-performance
  - Simplifying assumptions: humans are perfect, SW will eventually be bug free, HW MTBF is already very large and increasing, maintenance costs irrelevant to purchase price.
- New goals of systems research—addressing TCO
  - David Patterson, IPTS 2002: ACME "availability, change, maintenance, evolution"—total cost of ownership (TCO).
  - Jim Gray, HPTS 2001: FAASM "functionality, availability, agility, scalability, manageability"
  - Butler Lampson, SOSP 1999: "Always available, evolving while they run, growing without practical limit"
  - John Hennessy, FCRC 1999: "Availability, maintainability and ease of upgrades, scalability"

Recovery-Oriented Computing Philosophy

"If a problem has no solution, it may not be a problem, but a fact, not to be solved, but to be coped with over time"

— "Peres's Law"

- People, hardware, and software failures are facts, not problems.
- We cope with them through recovery/repair.
  - Recovery-Oriented Computing (ROC). Emphasizes recovery from failures rather than purely failure-avoidance.
- Improving recovery/repair improves availability
  - Availability = MTTF / (MTTF + MTTR)
  - Make MTTF very large; then Availability => 1, but, what if MTTR << MTTF
- ROC also helps with maintenance and TCO
  - Major system admin job is recovery after failure.
  - Since TCO is 5-10X HW/SW costs, spend extra on disk, DRAM, CPUs resources for recovery.
- More motivation—COTS have a fixed MTTF. Can only work with MTTR.

MTTR vs. MTTF

- Raising MTTF can never guarantee failure free operation.
  - But low MTTR could mitigate impact of failure.
  - Example: satellite tracking, antenna field of view, transient loss of antenna control.
- MTTF normally predicted vs. observed.
  - MTTF claims very difficult to verify directly .
  - MTTF doesn't capture end-user impact.
  - Do MTTR numbers include environmental error operator error, app bug?
  - Much easier to verify MTTR than MTTF!
- Lower MTTR may be strictly better than higher MTTF.
- Design goal: prevent outages and operate in a degraded state while attempting recovery.

ROC Infrastructure Mechanisms

- Recursive restartability—RR
  - Turning the reboot sledgehammer into a scalpel—minimize recovery time when using partial restarts to recover from transient failures.
  - Biggest improvement: MTTF/MTTR-based boundary redrawing of SW.
- Crash-Only Software
  - Only one way to stop, and only one way to bring up.
  - SW that crashes-safely and recovers quickly.
  - Recovery code is part of normal operation and therefore well-tested.
- Undo – at the system level
  - Time travel for system operators for high level commands
  - Three R's for recovery: rewind, repair, replay.
  - All three R's are critical: rewind enables undo, repair lets user/administrator fix problems, replay preserves updates, propagates fixes forward.
- Other work: path-based analysis, fault injection tools, online failure detection.

ROC in Ground Systems

- Composable ground stations—distributed GS components can be *composed* to form a *virtual ground station*.
  - A GS is decomposed into core components.
  - These are then assembled to form virtual ground station services.
  - Local teams for optimization, global teams for increased contacts.
- Ground Station Markup Language (GSML) – API for hierarchical command and control of typical ground station capabilities.
  - *Hardware Level* – Generic command of low level resources (ie radios).
  - *Session Level* – Services associated with single GS contacts. Sessions describe a space/ground communication channel specified over a specific time interval.
  - *Mission Level* -- Captures the services of a network of ground stations to support a space mission (handoffs, cooperative teaming on a pass).
  - Goal is acceptance of a design philosophy not necessarily GSML as a specific standard.
  - Built on XML messaging.

Testbeds

- Computer simulation testbed
  - Cluster of Linux PCs simulating space system.
  - Simple spacecraft simulators. Linux-based like our flight systems.
  - Simulated ground stations that run on single PCs (or VMs).
  - Beginning experiments on system level ROC techniques and GS composition.
- MGSN – The Mercury Ground Station Network
  - Networking global, university, low-cost OSCAR ground stations.
  - Supporting university satellites such as Stanford's OPAL and Sapphire. Also for the Cubesat program: 10-20 satellites a year.
  - Open source software available online for all interested organizations.
  - First node operational at Stanford and supports end-to-end IP access to space.
  - Deploying at ground stations in Germany and Norway and recruiting others.

Conclusions

- Research focus has changed in distributed/Internet computer systems from performance to focus on ACME- "availability, change, maintenance, evolution".
- Space systems are becoming more Internet-like in nature with similar design requirements, challenges, and components.
- We're in the process of applying recovery oriented computing principles to space systems, focusing now on ground systems.
- Additional information
  - http://swig.stanford.edu/
  - http://ssdl.stanford.edu/
  - http://www.mgsn.net/

- Extra Slides

Five "ROC Solid" Principles

1. Given errors occur, design to recover rapidly
2. Given humans make errors, build tools to help operator find and repair problems
   - e.g., undo; hot swap; graceful, gradual SW upgrade
3. Extensive sanity checks during operation
   - To discover failures quickly (and to help debug)
   - Report to operator (and remotely to developers)
4. Any error message in HW or SW can be routinely invoked, scripted for regression test
   - To test emergency routines during development
   - To validate emergency routines in field
   - To train operators in field
5. Recovery benchmarks to measure progress
   - Recreate performance benchmark competition

From D. Patterson talk on ROC at UIUC

Traditional Fault Tolerance vs. ROC

- >30 years of Fault-Tolerance research
  - fewer systems builders involved; ROC is for/by systems builders
- FT greatest success in HW; ignores operator error?
  - ROC holistic, all failure sources: HW, SW, and operator
- Key FT approach: assumes accurate model of HW and SW, and ways HW and SW can fail
  - Models to design, evaluate availability
  - Systems/ROC: benchmarks, quantitative evaluation of prototypes
- Success areas for FT: airplanes, satellites, space shuttle, telecommunications, finance (Tandem)
  - Hardware, software often changes slowly
  - Where SW/HW changes more rapidly, less impact of FT research
- Much of FT helps MTTF, ROC helps MTTR
  - Improving MTTF and MTTR synergistic (don't want bad MTTF!)

From D. Patterson talk on ROC at UIUC

Lessons of Internet Services

Internet services programmed with a "bunker mentality"

1. Preserve fault isolation boundaries
   - Containment--exploit natural isolation boundaries to contain faults (clusters,virtual machines)
2. Explicitly encapsulate state
   - Protection—all state in a well-known, protected place (HTTP)
3. Separate data format from implementation
   - Versatility—data exchange is independent of transport (HTML over HTTP, WAP, etc.)
4. Orthogonal checks and monitors
   - Reliability—component level and end-to-end checks
5. Design for restartability
   - Recovery—improving availability through lower MTTR and rejuvenation

Direct Downtime Costs (per Hour)

Brokerage operations	$6,450,000
Credit card authorization	$2,600,000
Ebay (22 hour outage)	$225,000
Amazon.com	$180,000
Package shipping services	$150,000
Home shopping channel	$113,000
Catalog sales center	$90,000
Airline reservation center	$89,000
Cellular service activation	$41,000
On-line network fees	$25,000
ATM service fees	$14,000

Sources: InternetWeek 4/3/2000 + *Fibre Channel: A Comprehensive Introduction*, R. Kembel 2000, p.8. "...based on a survey done by Contingency Planning Research."

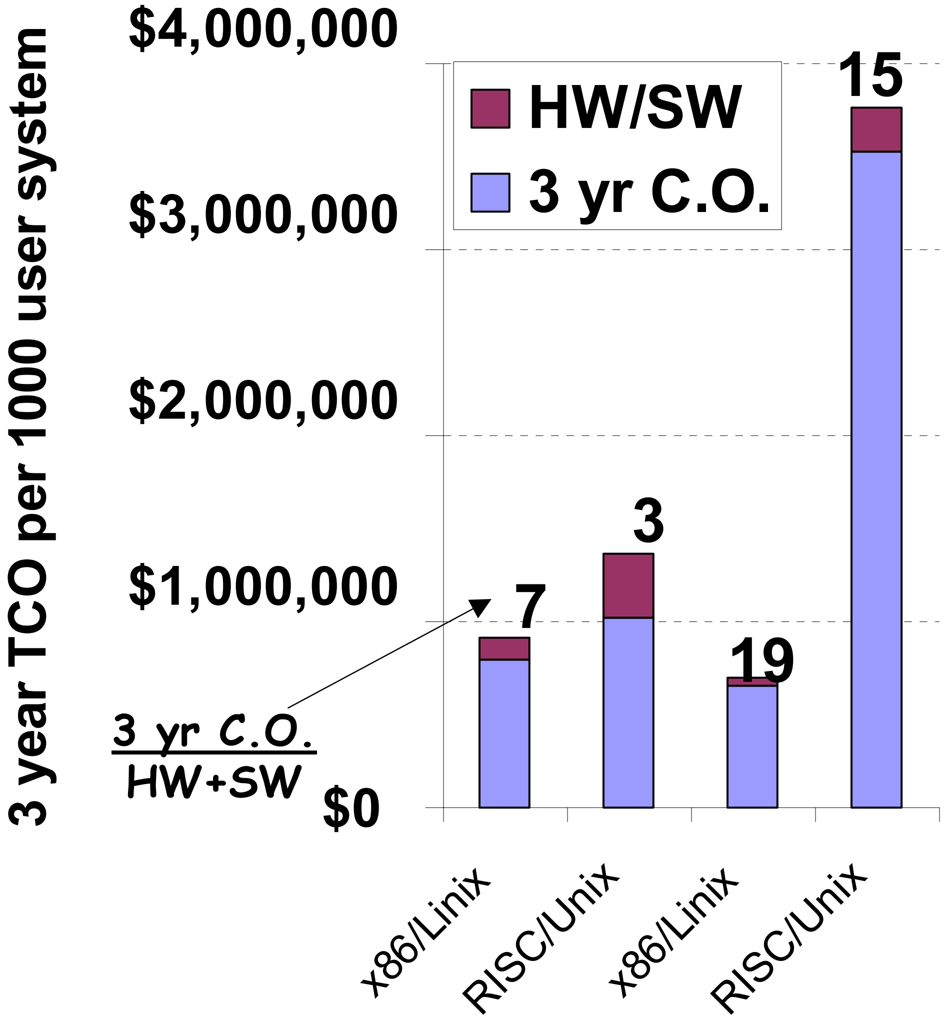
The 3R undo model

- Undo == time travel for system operators
- Three R's for recovery
  - **Rewind:** roll system state backwards in time
  - **Repair:** change system to prevent failure
    - e.g., edit history, fix latent error, retry unsuccessful operation, install preventative patch
  - **Replay:** roll system state forward, replaying end-user interactions lost during rewind
- All three R's are critical
  - rewind enables undo
  - repair lets user/administrator fix problems
  - replay preserves updates, propagates fixes forward

Virtual Machine Monitors

- Goal: explicit fault isolation boundaries. Prevent errors from propagating.
- Virtual machines as an isolation mechanism:
  - Examples: JVM's, Vmware
  - Provide isolation comparable to physical hardware separation.
  - Reservation of critical resources for disaster recovery.
  - VM's monitorable for introspection [Noble & Chen, 2001].
- Why do we trust VMM's?
  - Simpler than the underlying OS with more narrow interfaces.
  - They rely largely on even simpler underlying HW mechanisms (hardware timers, hardware virtual memory mgt, etc.).
  - We trust those underlying HW mechanisms because: even simpler and orthogonal to OS, implemented in HW, extensively tested, low churn on implementation.

Total Cost of Ownership

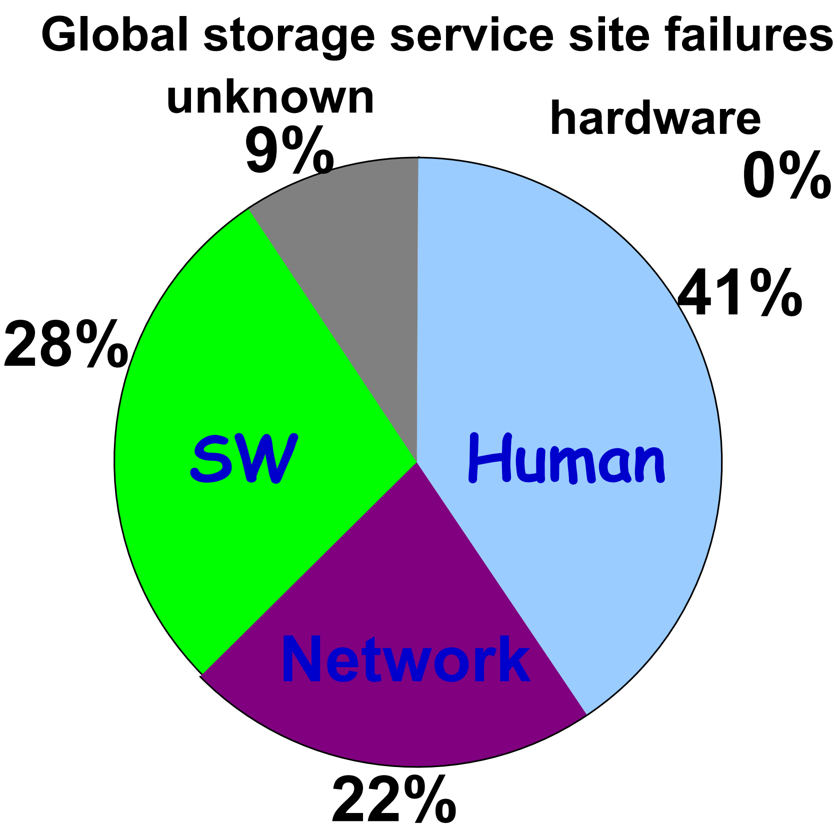


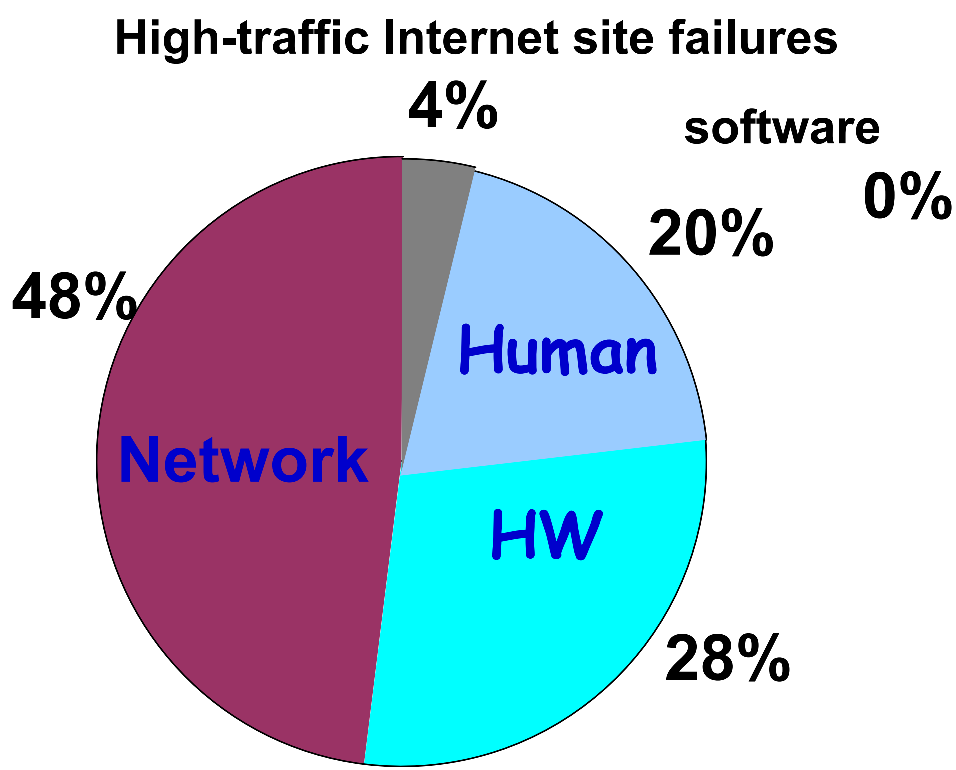
- 142 Interviews, 2H01
- $2.4B/yr avg. sales
- Avg. 3 - 12 servers, 1100 - 7600 users/site
- not included: space, power, media, comm., HW/SW support contracts, downtime
- Internet/Intranet: firewall,Web serving, Web caching, B2B, B2C
- Collaborative: calendar, email, file/database

From D. Patterson talk on ROC at UIUC

Source: "The Role of Linux in Reducing the Cost of Enterprise Computing", IDC white paper, sponsored by Red Hat, by Al Gillen, Dan Kusnetzky, and Scott McLaron, Jan. 2002, available at www.redhat.com

Internet System Failures



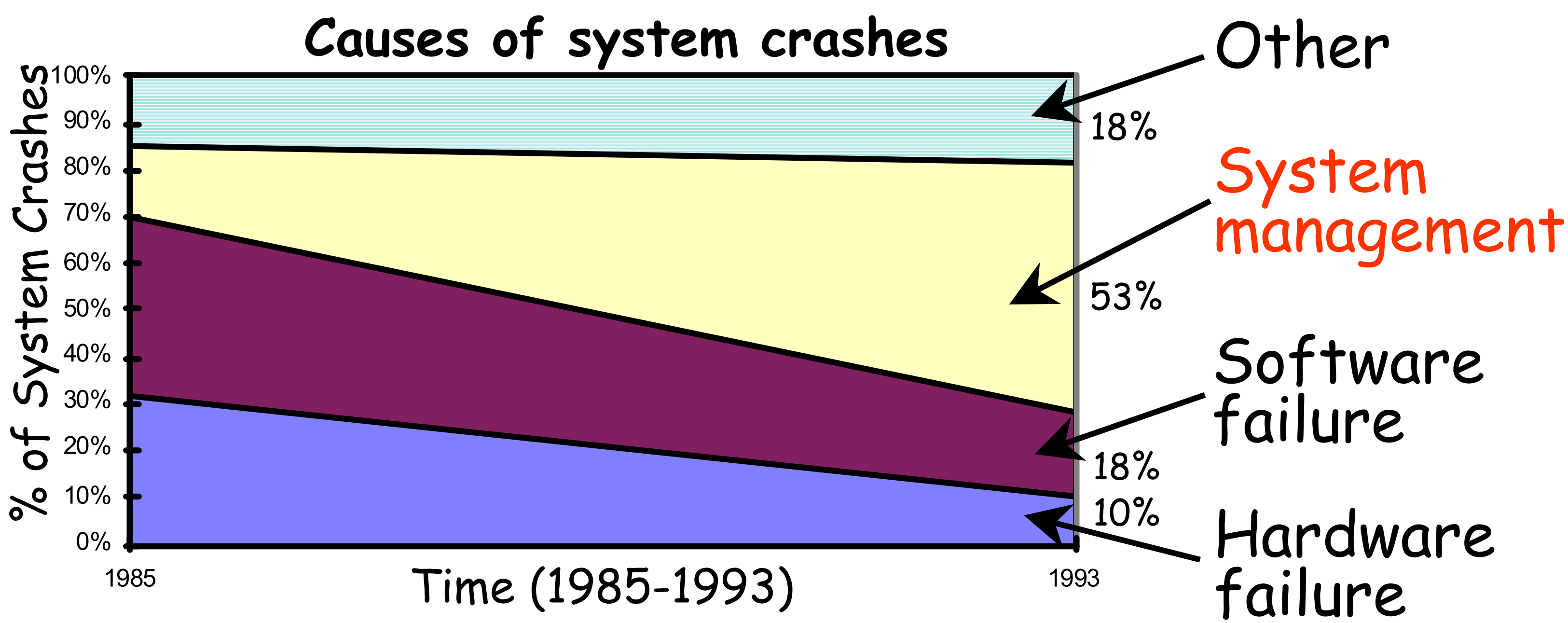


- Human error largest cause of failure in the more complex service, significant in both
- Network problems largest cause of failure in the less complex service, significant in both

From D. Patterson talk on ROC at UIUC

Lessons About Human Operators

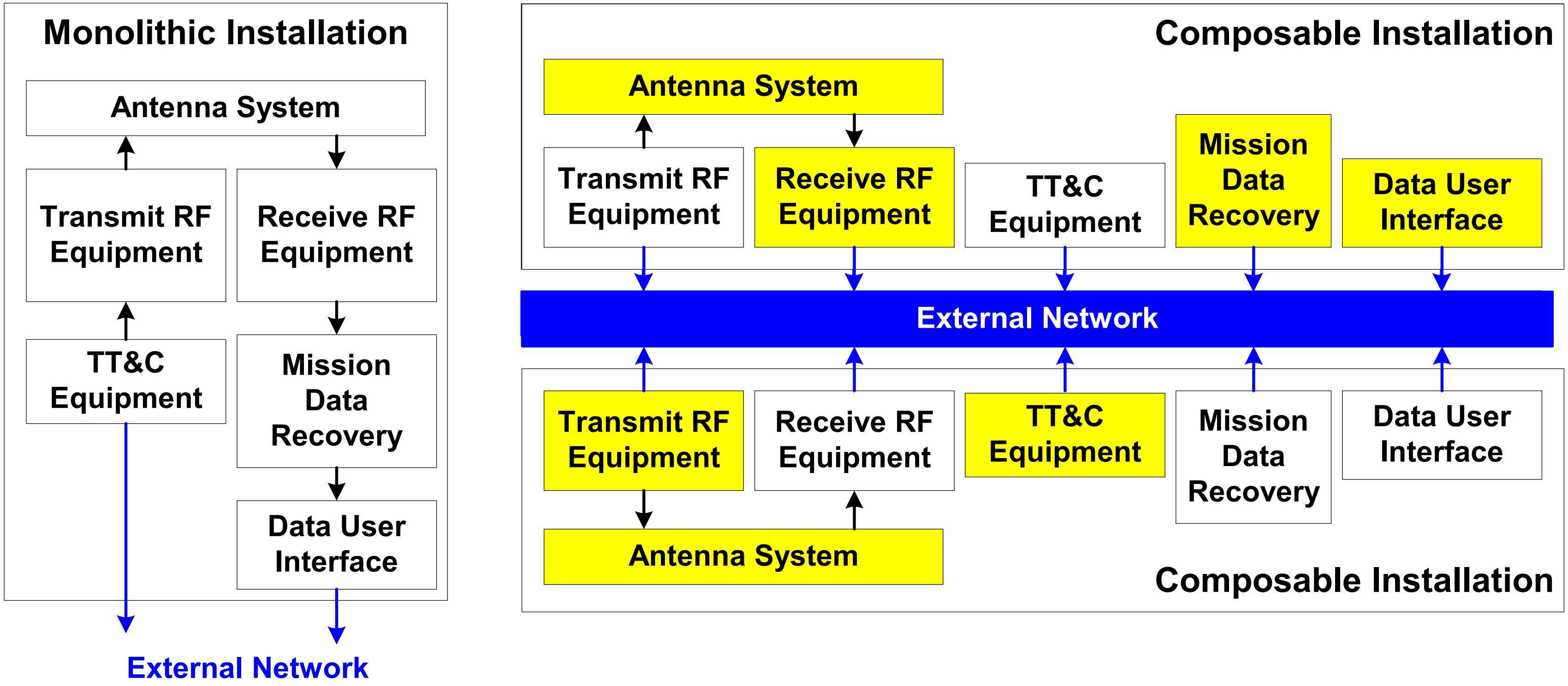
- Human error is largest single failure source
  - HP HA labs: human error is #1 cause of failures (2001)
  - Oracle: half of DB failures due to human error (1999)
  - Gray/Tandem: 42% of failures from human administrator errors (1986)
  - Murphy/Gent study of VAX systems (1993)



From D. Patterson talk on ROC at UIUC

Composable GS

- Distributed GS components can be *composed* to form a *virtual ground station*.
  - A GS is decomposed into core components.
  - These are then assembled to form virtual ground station services.
  - Local teams for optimization, global teams for increased contacts.



Mercury Architecture