

COE CST Seventh Annual Technical Meeting

Task #319. DebrisSat Panel Preparation and Fragment Characterization for the Period: FY17 Q3

**Norman Fitz-Coy
Joe Kleespies**

*October 10, 2017
Las Cruces, NM*

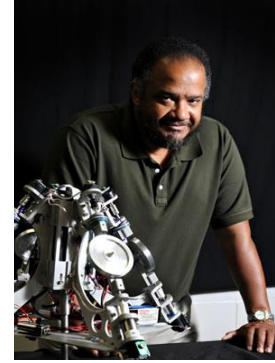


Agenda

- Team Members
- Task Description
- Schedule
- Goals
- Task Discussion
- Conclusions and Future Work

Team Members

- People
 - Norman Fitz-Coy (PI)
 - Joe Kleespies (Grad Student)
- Organizations



Task Description

- Objectives:
 - Implement “big data” management solution and procedures for archiving DebrisSat’s data.
 - Ensure all project data management requirements are satisfied.
 - Evaluate and optimize the performance of the data management solution.
 - Facilitate the transfer of DebrisSat data to the project stakeholders.

Schedule

Semester	Tasks
Fall 2016	<ul style="list-style-type: none"> • Research viable database engines and storage methods. • Install and configure new database engine. • Define and document structure of new database engine and subsequent relational tables. • Begin modification of the existing DCS front-end layer.
Spring 2017	<ul style="list-style-type: none"> • Complete modification of the existing DCS front-end layer. • Implement new image and file storage structure. • Begin addition of “3D” imaging system fields and formats.
Summer 2017	<ul style="list-style-type: none"> • Evaluate and optimize database engine performance.
Fall 2017	<ul style="list-style-type: none"> • Complete addition of “3D” imaging system fields and formats. • Begin documentation of upgrade process and maintenance protocols.
Spring 2018	<ul style="list-style-type: none"> • Complete documentation of upgrade process and maintenance protocols.

Goals

- Outcomes:
 - Database solution used for the data management for DebrisSat project.
 - Database solution used for data management for similar “big data” projects.
- Relevance to FAA:
 - Orbital debris modeling is critical to achieving better space traffic management.
 - Archival database using “big data” framework for improved modeling of space debris.

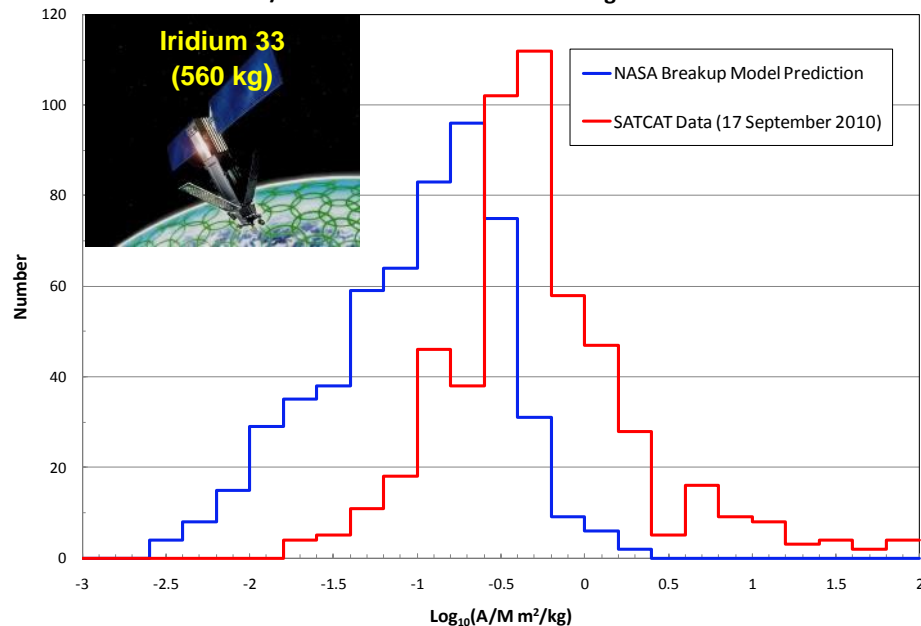
Task Discussion



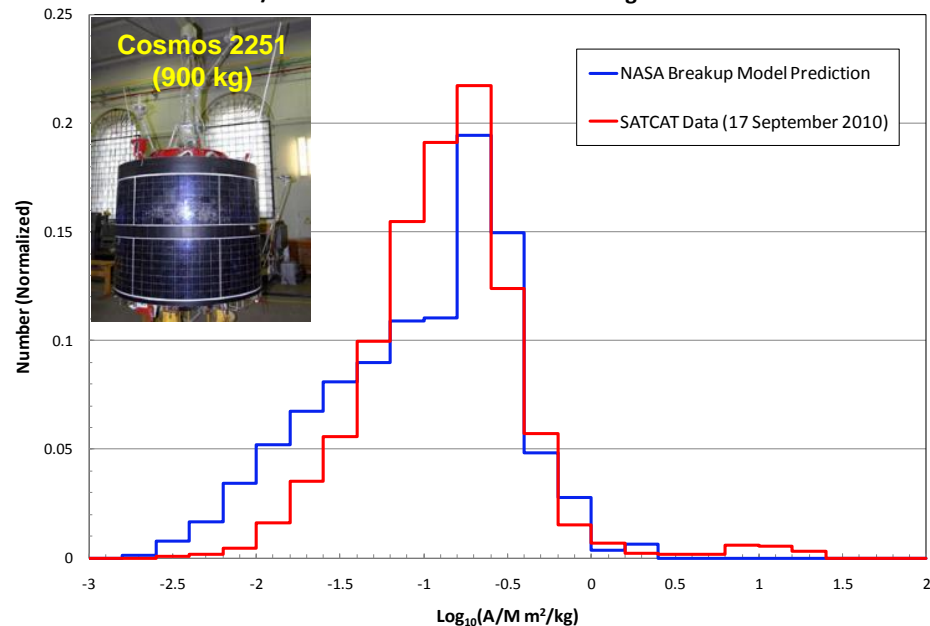
Orbital Debris Background

- 23,000+ objects, Iridium 33 – Cosmos 2251 collision was a big contributor.
- Current satellite breakup models based on the SOCIT series of laboratory hyper-velocity impact (HVI) tests.
- Existing models work well for old satellites, less so for newer satellites with modern materials and processes.

A/M Distribution of Iridium 33 Fragments

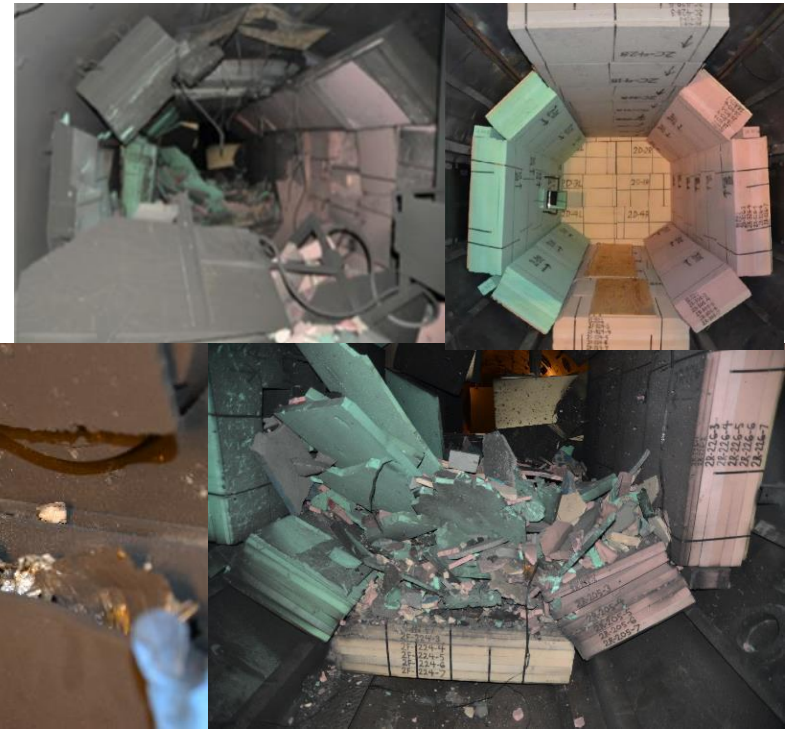
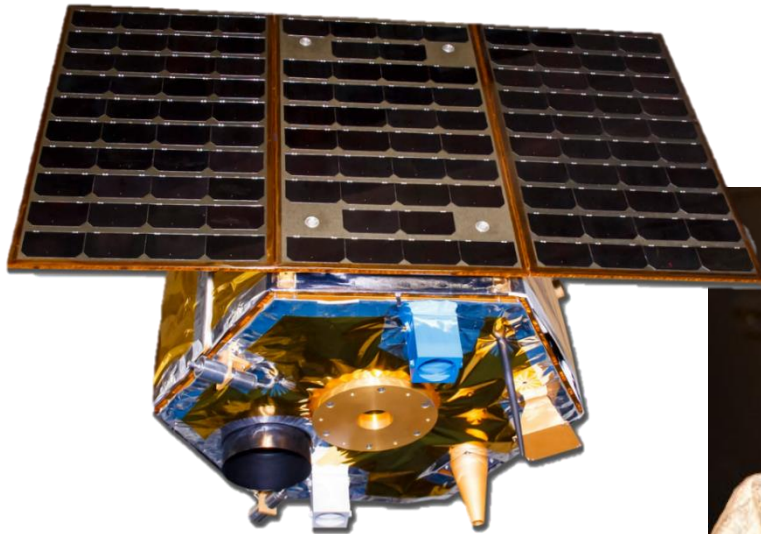


A/M Distribution of Cosmos 2251 Fragments



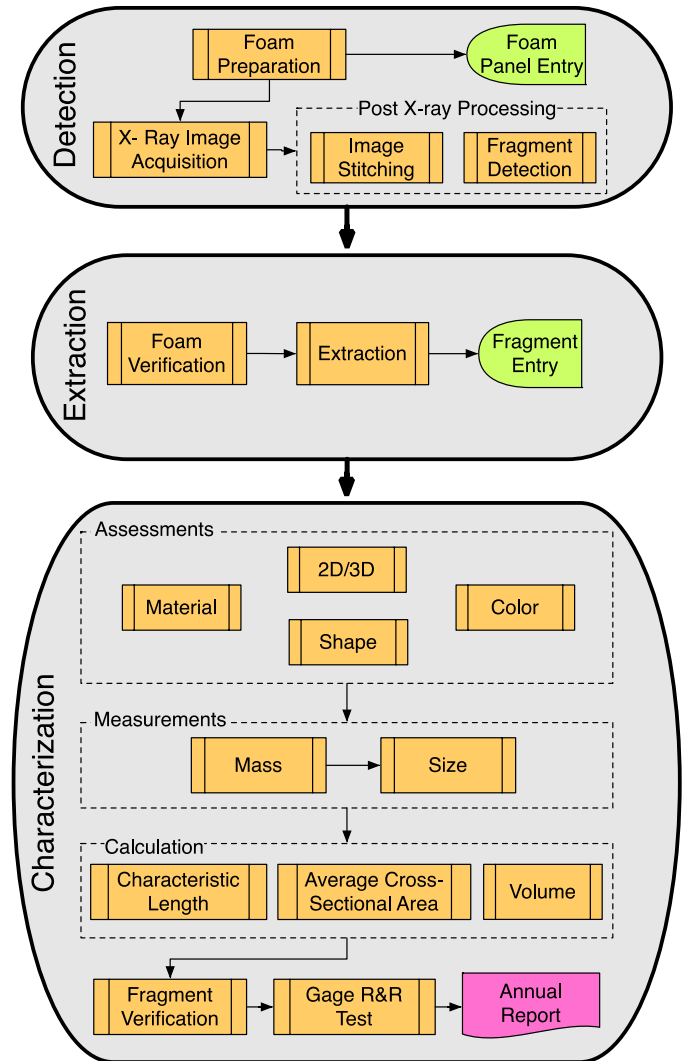
DebrisSat Overview

- Test article designed and fabricated as a “representative” modern LEO satellite using modern materials and processes.
- Goal: Update existing satellite breakup models.
- Laboratory HVI test performed in April 2014.
 - 56 kg representative LEO satellite
 - Impact speed of 6.8 m/s (13.2 MJ)



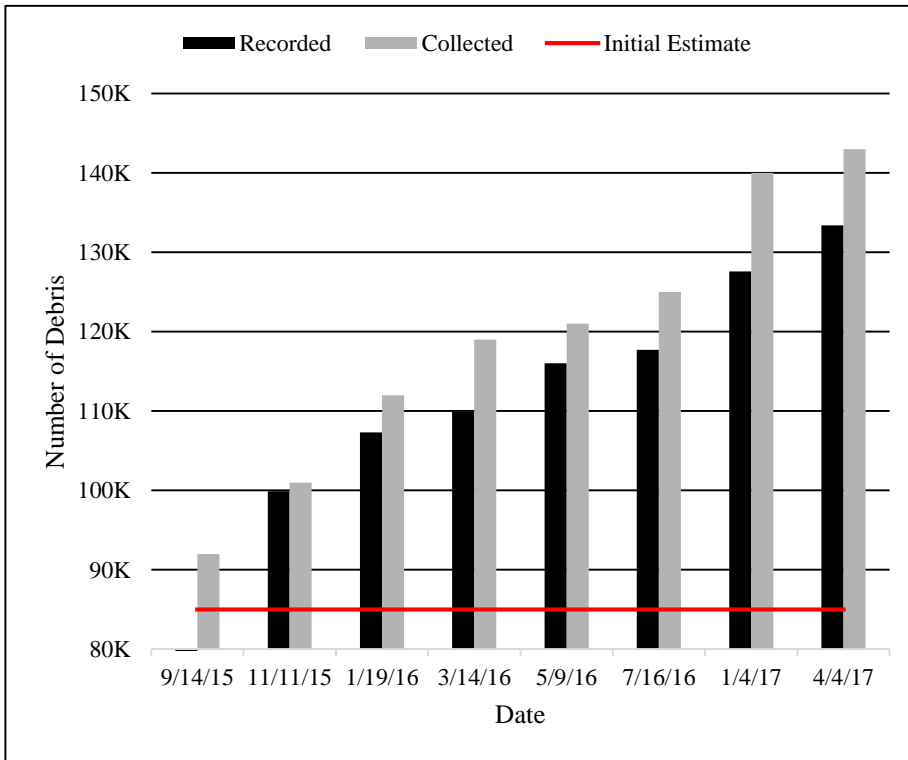
Post-HVI Process

- Fragment Detection:
 - Prepare foam panels for X-ray imaging.
 - Process X-ray images to detect embedded fragments.
- Fragment Extraction:
 - Excavate fragments ≥ 2 mm in one dimension.
- Fragment Characterization:
 - Assess fragment's physical attributes (2D/3D, material, shape, and color).
 - Measure fragment's mass and sizes.
 - Archive all fragment data, images, and associated metadata in database.
 - Verify fragment's database entry.



Data Management Challenge

- Originally expected 85,000 debris fragments.
- Collected and recorded over 200,000 fragments to date (and still counting).
- 300+ data fields utilized for each debris fragment.



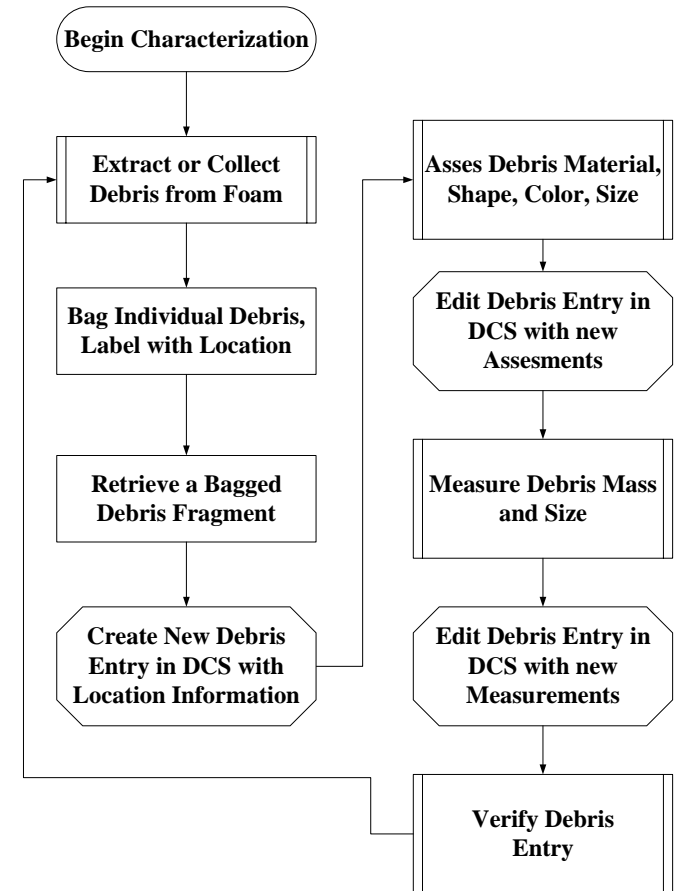
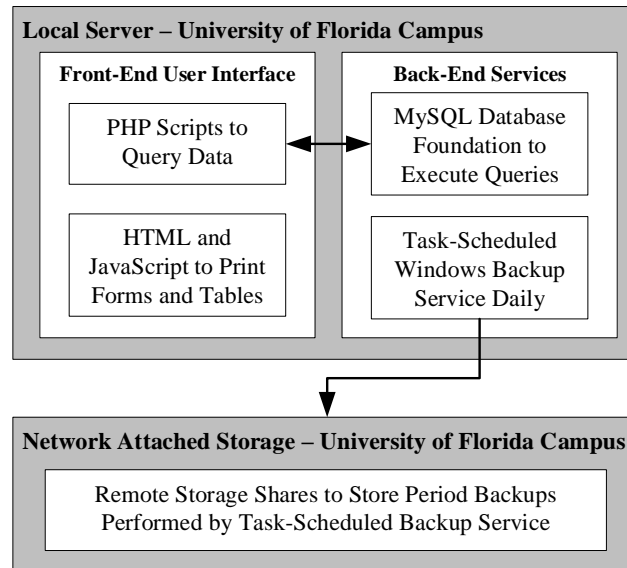
<i>index</i>	<i>section</i>	<i>imagerSoftwareVersion</i>
<i>project</i>	<i>row</i>	<i>mass_g</i>
<i>debrisId</i>	<i>area</i>	<i>temperature_C</i>
<i>revisionNumber</i>	<i>foamGridLocation</i>	<i>humidity_%</i>
<i>currentRevision</i>	<i>primaryMaterial</i>	<i>xDimension_mm</i>
<i>boxNumber</i>	<i>secondMaterial</i>	<i>yDimension_mm</i>
<i>creator</i>	<i>thirdMaterial</i>	<i>zDimension_mm</i>
<i>creationTimestamp</i>	<i>shape</i>	<i>characteristicLength_mm</i>
<i>verified</i>	<i>identifyingColor</i>	<i>volume_mm^3</i>
<i>verifier</i>	<i>debrisType</i>	<i>density_g/mm^3</i>
<i>verificationTimestamp</i>	<i>foamAttached</i>	<i>acsa_mm^2</i>
<i>source</i>	<i>intactPart</i>	<i>amr_mm^2/g</i>
<i>relatedFoam</i>	<i>balanceType</i>	<i>frontlitCapture</i>
<i>foamId</i>	<i>balanceSoftwareVersion</i>	<i>backlitCapture</i>
<i>foamType</i>	<i>imagerType</i>	...

Debris Categorization System

- Requirements:
 - Record data from foam and debris.
 - Guide user through post-impact process.
 - Minimize human error.
 - Maximize user efficiency.
 - Ensure data perpetuity and integrity.

Front-end, back-end dichotomy.

Designed and developed in parallel with the post-HVI phase procedures.



Phases of DCS Implementation

- Two Phases:
 - *Rapid-Development Phase (initial)*
 - Get the system online and operational as fast as possible.
 - Respond to frequent changes in requirements, data fields, etc.
 - *Operations Phase (current)*
 - Ensure all the project requirements are satisfied.
 - Ensure data perpetuity and integrity.
 - Make the system as maintenance-free as possible.
 - Facilitate data transfer to stakeholder organizations.



Rapid-Development Phase

- Simple, convenient, quick design choices made.
 - MySQL database engine used for simplicity.
 - Rich data stored as dynamic links.
- Frequency changes to requirements, data fields and processes mirrored in post-HVI workflow.
- Functional front-end interface and back-end services implemented in 6 months.

Fragment Mass Measurement

Mass (g)	Temperature (C)	Humidity (%)
0.000006	24.10	51.20

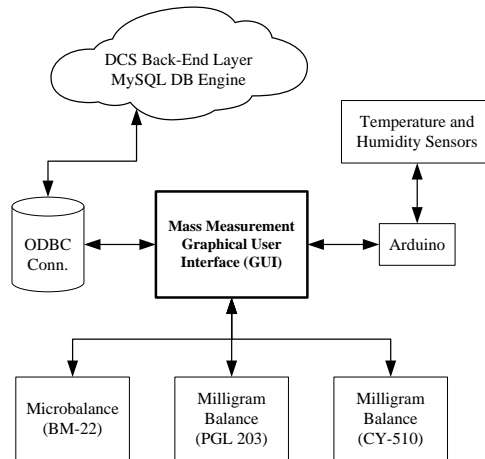
Mass Balance: Micro Mass

Instructions

Mass measured. Re-mass boat. If +/- 0.000010g, click Upload, else weigh again

Gatorlink: camila96

Debris ID: DS112152



IDENTIFICATION
Project: DebrisSat Box #:

LOCATION
Source: ▼ Related Foam:
Section: ▼

ASSESSMENT
Primary Material: ▼ Second Material: NONE Third Material: NONE
Shape: ▼ Ident. Color: ▼ Debris Type: 20
Foam Attached: Intact Part:

MEASUREMENT SYSTEMS
Balance Type: ▼ Balance Software Version:
Imager Type: ▼ Imager Software Version:

MEASUREMENTS
Mass: g Temp: °C RH%: %
X_{DIM}: mm Y_{DIM}: mm Z_{DIM}: mm
L_C: mm Volume: mm³ Density: g/mm³
ACSA: mm² AMR: mm²/g

IMAGING CAPTURES
Frontlit Capture: No file chosen
Backlit Capture: No file chosen

IMAGING ANALYSES
Height Detection: No file chosen
Edge Detection: No file chosen
Ring Calibration: No file chosen
Debris Analysis: No file chosen

IMAGING DATA
2D Point Cloud: No file chosen
Region Properties: No file chosen

COMMENTS

Operations Phase

- Primary Goal: Ensure data perpetuity and integrity (i.e. ACID-compliant).
 - ACID (Atomicity, Consistency, Isolation, Durability) used to describe the validity of database engine transactions.
- Indirect storage method for rich data implemented in rapid-development phase posed a risk to data perpetuity and integrity.

	Pros	Cons
Direct Storage - Guaranteed perpetuity of entire dataset - Increased data integrity - Faster performance for some access patterns - Transfer of full dataset is easier and quicker, requires less attention	Indirect Storage	- Longer database-wide operations and queries - More complex querying (e.g. get a range of fragments greater than 0.5 TB) - Faster access via filesystem - Transfer of full dataset is easier and quicker, requires less attention - Access outside of database

- Decision (requirements-driven) to implement direct BLOB storage.

BLOB Storage in Industry

- Paper from Microsoft Research in 2006 concluded for files larger than 1 MB, indirect storage was better; for files less than 256 kb, direct storage was better; in between it depends on the application.

Facebook: Haystack and f4 warm BLOB storage (hybrid)

Twitter: Blobstore (indirect)

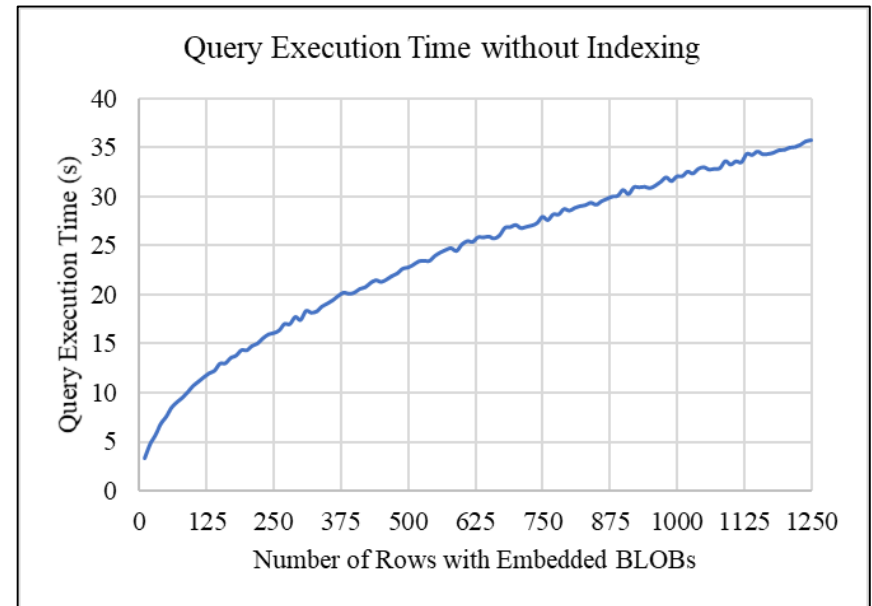
Azure: BLOB storage provider, different types of BLOBs (hybrid)

SharePoint: Storage bins and FILESTREAM (direct)

- Storage method decisions are requirements-driven.
- Almost all of the major companies use custom, hybrid solutions.
- Facebook and Twitter don't necessarily require data perpetuity.

Performance Concerns

- Direct BLOB storage and incur significant query performance impacts.
- Example:
 - Copied data from between *rapid-development* version to *production* version.
 - Exponential-like query execution time while inserting data.
 - Time to finish copy became prohibitively high.

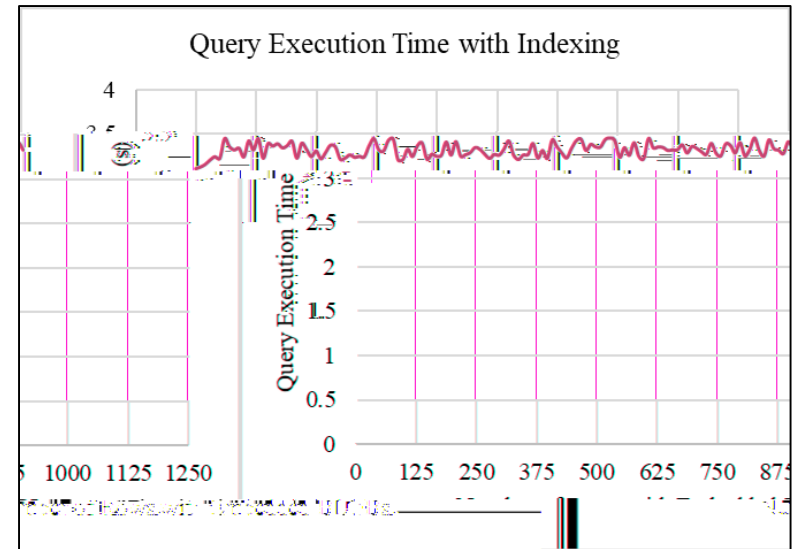


Performance Solutions

- After analyzing the database engine status, noticed a majority of query execution time was consumed by full database-wide scan for rows.
 - Column indexing eliminates full-database scans for queries using indexed columns in their WHERE clauses.
- Front-end user interface could be optimized by writing clever queries leveraging structures like INNER JOIN and avoiding database-wide scans.

Consistently low query execution time for same data transfer example after column indexing.

Use of clever SQL “tricks” can be used to mitigate negative performance impact from direct BLOB storage.



Conclusions and Future Work

- Debris Categorization System (DCS) was designed, developed, and implemented in two distinct stages with different goals.
- Rapid-development phase was necessary because the requirements and structure of the DCS changed frequently and mirrored changes in the physical post-impact phase procedures.
- The goals of the operations phase were primarily to address the shortcomings of quick and convenient design choices in the rapid-development phase.
- Direct BLOB storage was chosen to ensure data perpetuity and integrity per the requirements of the DebrisSat project.
- Various SQL “tricks” were employed to mitigate the negative performance impact of direct BLOB storage.