PORTICO

PORTICO

Digital Content Management at Scale: A Case Study from Portico

Evan Owens, Chief Technology Officer
Vinay Cheruku, Technical Lead, Systems
John Meyer, Technical Lead, Data and Tools
Sheila Morrissey, Senior Research Developer

www.portico.org

PORTICO

This Presentation

Scale Matters

Scale is a mysterious phenomenon -- processes that work fine at one scale can fail at 10 times that size, and processes that successfully handle a 10-times scale can fail at 100 times. [...] Institutions offering tools and systems for digital preservation should be careful to explain the scale(s) at which their systems have been tested, and institutions implementing such systems should ideally test them at scales far above their intended daily operation, probably using dummy data, in order to have a sense of when scaling issues are likely to appear.

Clay Shirkey, Library of Congress Archive Ingest and Handling Test (AIHT) Final Report, June 2005, page 26
http://www.digitalpreservation.gov/library/pdf/ndiipp_aiht_final_report.pdf

PART I

The Context

Portico Business Summary

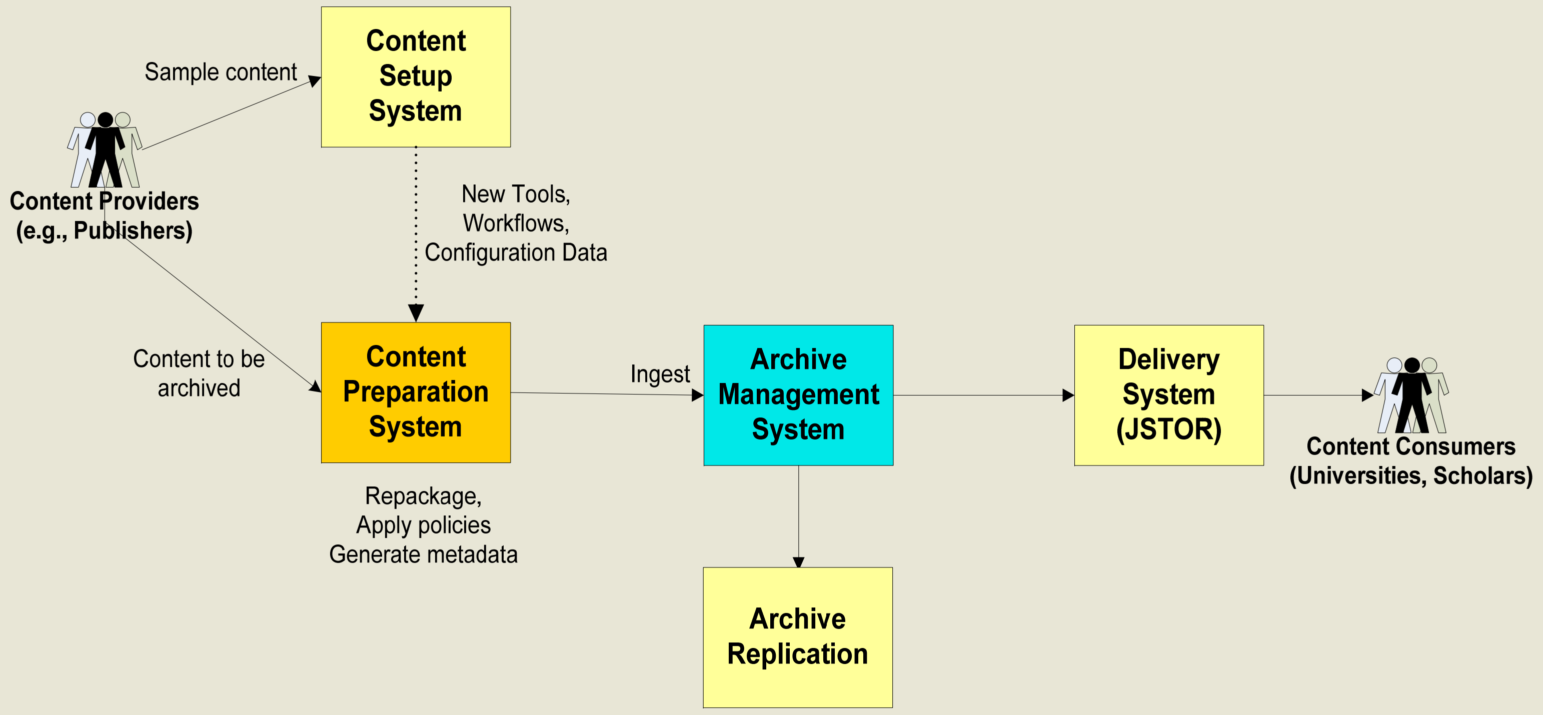
- A permanent archive of scholarly literature in electronic form
  - All preservation and access rights secured by irrevocable contractual agreements
- Start-up funding by Andrew W. Mellon Foundation, JSTOR, Ithaka, and Library of Congress NDIIPP
- Initial content area is E-Journals
- E-Journal statistics as of late April 2008:
  - 54 participating publishers
  - 7,596 journal titles committed
  - 433 participating libraries from 11 countries
  - 7,048,038 articles archived (of >14M articles committed)
    - > 70 million files
    - 94 file formats
  - Eventual size of E-Journal collection
    - ~ 200 million files
    - 18-20 million articles

PORTICO

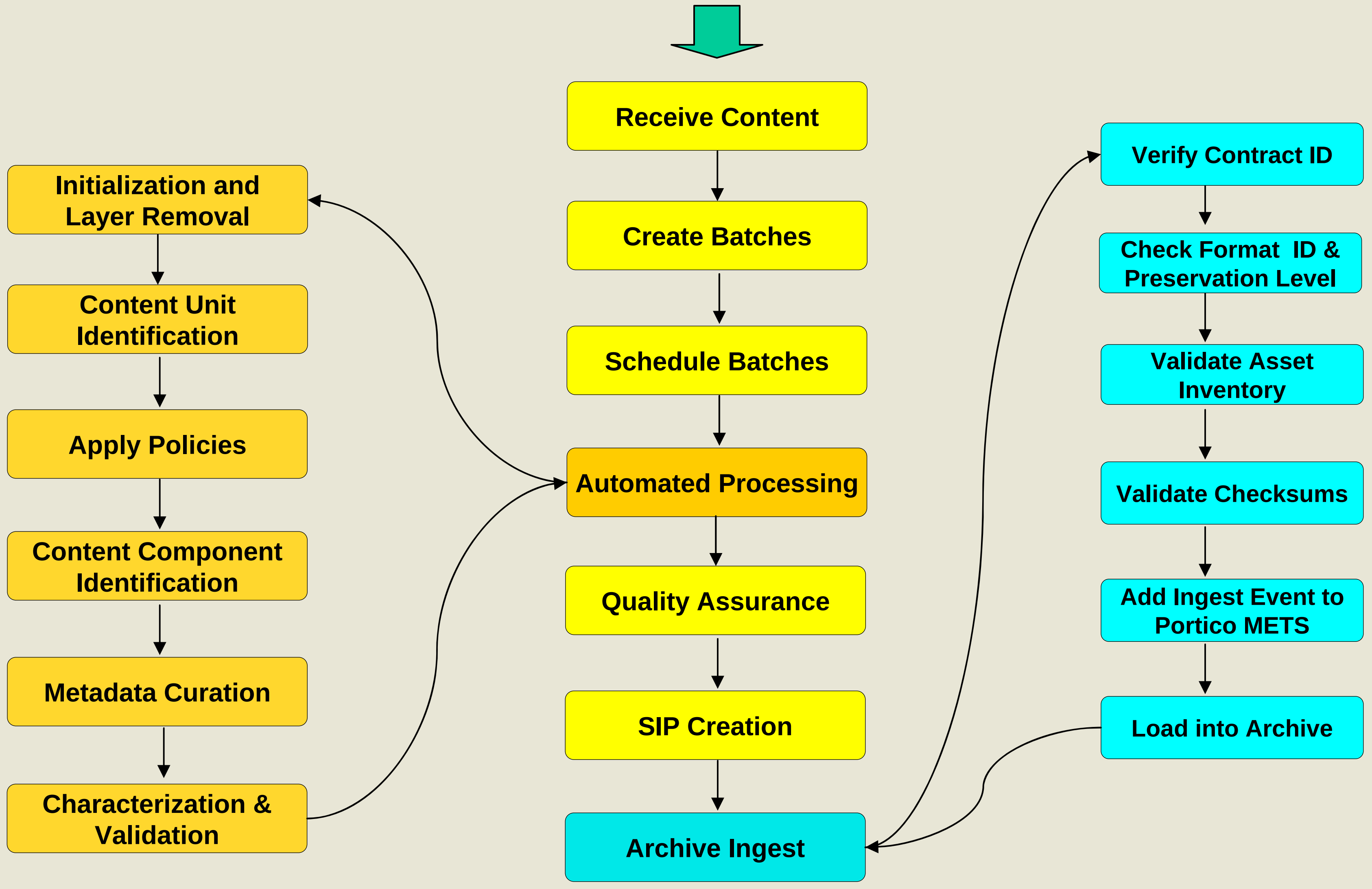
Portico Technology Summary

- OAIS-compliant repository designed for managed preservation
- Key influences:
  - OAIS, GDFR, PreMIS, METS, MPEG-21, ARK
- Key technologies:
  - XML, XML schema, Schematron, JHOVE, NOID
  - Documentum, Oracle, Java, JMS, LDAP
  - Format Registry
- Archive design goals:
  - Content preserved in application-neutral content using open standards
    - METS, PREMIS, JHOVE
  - A “Bootstrapable Archive”
    - XML plus digital objects
- Ingest system design goals:
  - Pluggable tools to facilitate new providers and replacement tools
  - Configurable workflows for different content types
  - Scalable to very high content volumes…*in theory and now in practice*

Portico Systems Overview


Content Setup System
Sample content
Content Providers (e.g., Publishers)
New Tools, Workflows, Configuration Data
Content to be archived
Content Preparation System
Repackage, Apply policies Generate metadata
Ingest
Archive Management System
Delivery System (JSTOR)
Content Consumers (Universities, Scholars)
Archive Replication

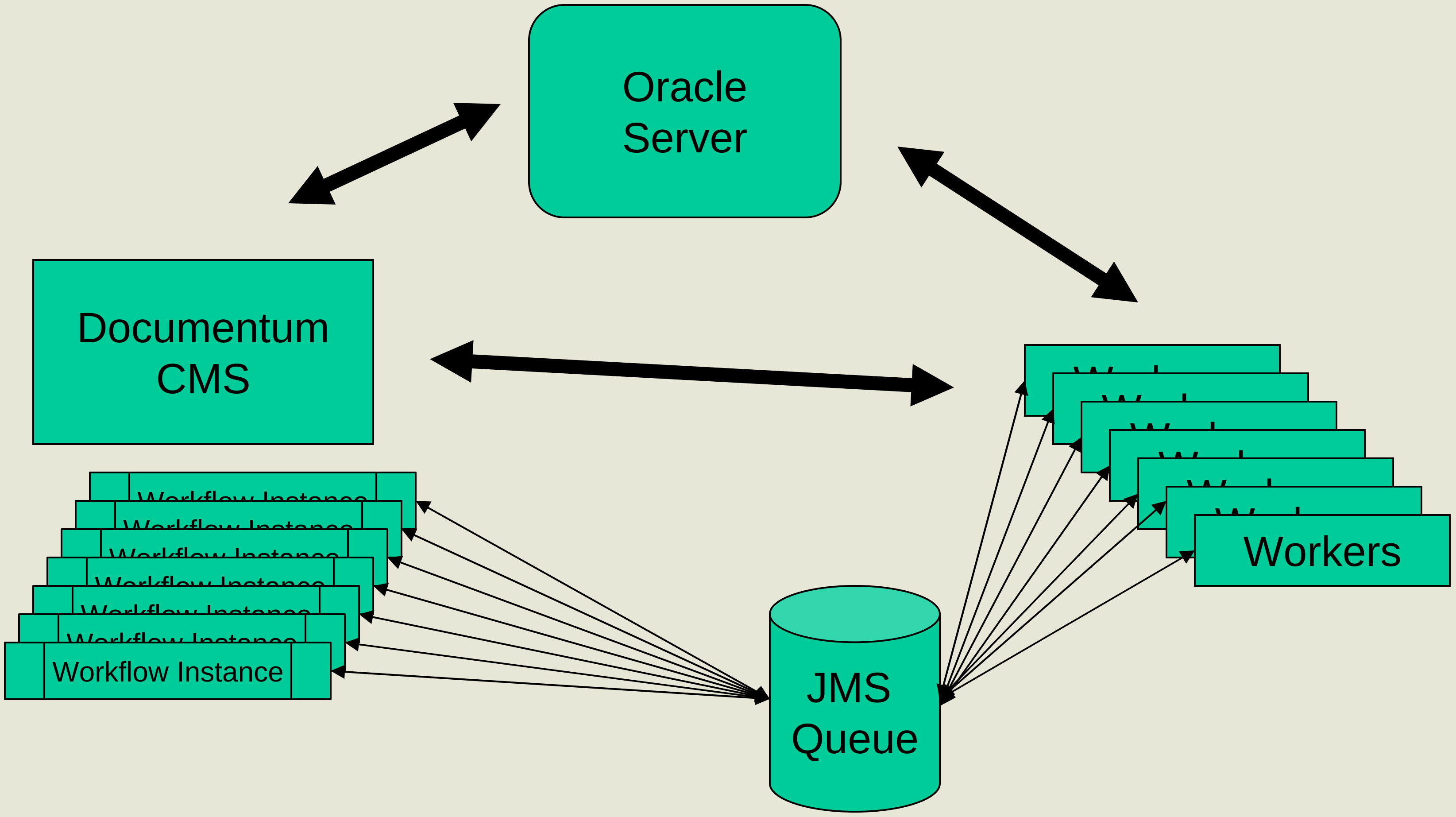
PORTICO


Receive Content
Create Batches
Schedule Batches
Automated Processing
Quality Assurance
SIP Creation
Archive Ingest
Initialization and Layer Removal
Content Unit Identification
Apply Policies
Content Component Identification
Metadata Curation
Characterization & Validation
Verify Contract ID
Check Format ID & Preservation Level
Validate Asset Inventory
Validate Checksums
Add Ingest Event to Portico METS
Load into Archive

Portico Content Processing

- Inputs
  - Per unit of content (e.g., article):
    - one text or metadata file, zero or more other files
  - Arbitrary (provider-specific) collections of data
    - Standard or proprietary file & directory naming conventions
    - Standard or proprietary formats
  - Undocumented business rules hidden in the data
- Outputs
  - Content packaged in Portico METS
  - Metadata: technical, descriptive, events
    - ~100 GB of metadata for every 1 TB of content
    - E-Journal content averages 10 million files per 1 TB
  - Content restructured to Portico content model
    - Article component structure documented
  - Content normalized as per preservation plans
    - Proprietary publisher formats converted to NLM Archival DTD
    - PDF created from TIFF as needed
- Processing Requirements
  - Vary according to formats and number of files in batch

High-Level Content Preparation System Overview



PORTICO

PART II
The Challenge of Scaling

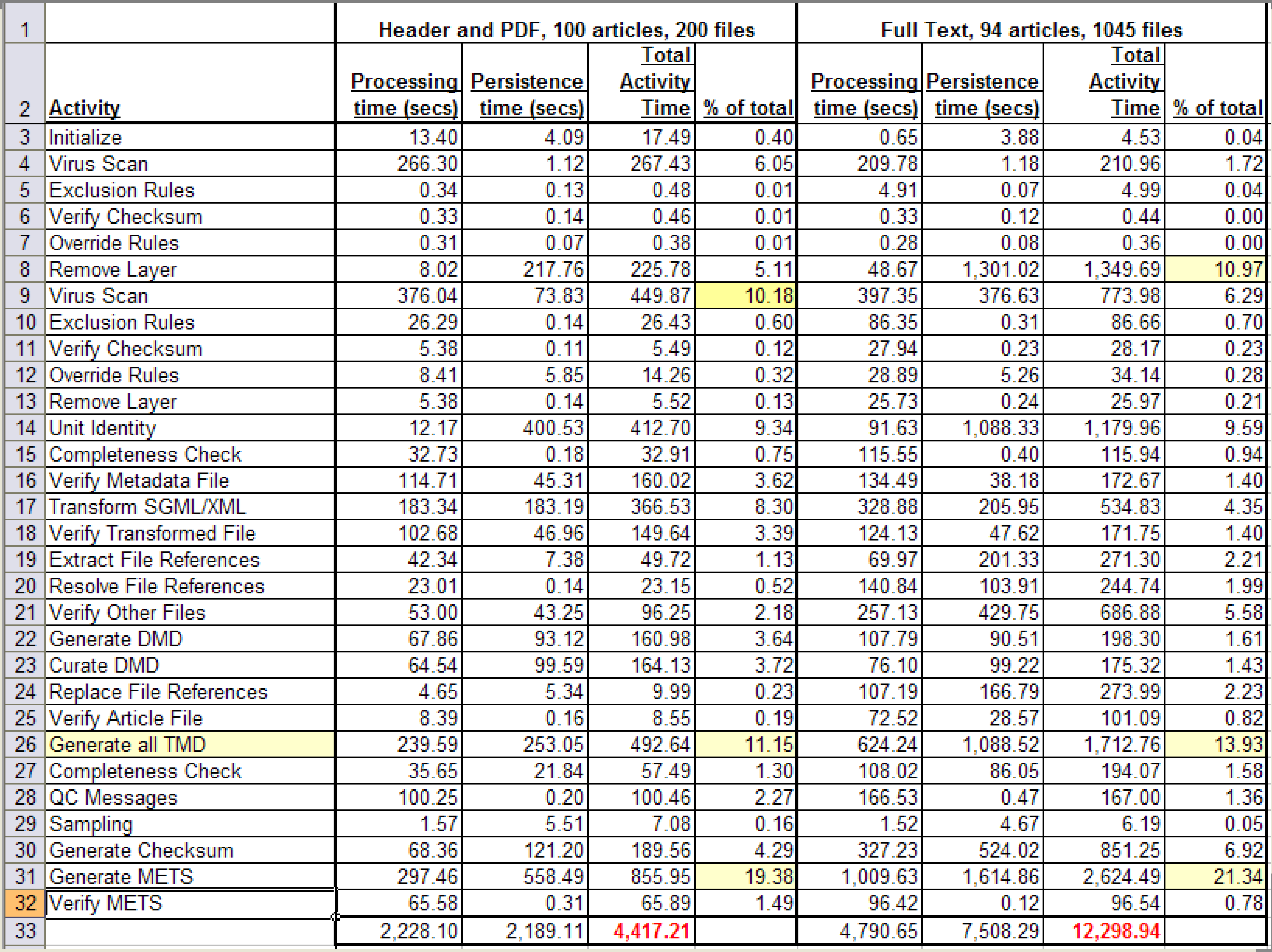
Time Travel: Back to August-September 2006

- A note about capacity calculations
  - Typical header + PDF = 2 files, 1 article
  - Typical XML full text + PDF = 10+ files, 1 article
  - Count articles? Count files? Count Bytes?
- Content Processing System status as of 9/06
  - Live since March 2006
  - System capacity around 75K articles / month on existing hardware
  - Experience to date only about 126K articles (700K files) ingested
  - A prototype moved into production
  - In theory scalable but hitting performance roadblocks
- Business status as of 9/06
  - Signing publishers faster than expected
  - Signing more large publishers than expected
- Business goals for 2007
  - 4 to 6 million new articles (40 to 60 million files) ingested into the archive
  - Support new content types and increased capacity in 2008
- Technology goal for 2007
  - Increase total capacity to 10M articles / 100M files per year

How to Scale the System?

- Hardware
  - Hardware only solution not an option due to design limitations in persistence model
  - Design allowed for multiple workflow listener machines
  - More hardware? How much more?
  - Faster hardware? Where and why?
- Software
  - We knew that there was a serious performance bottleneck in persistence
  - How bad was it? Could we fix it?
  - What else needed to be improved? What to leave for now?
- Balancing Act
  - Cost of hardware versus cost of software development
- What should our approach be?
  - Raise the speed limit?
  - Add more lanes to the highway?
  - Reduce traffic jams?
- We needed data
  - Testing to the rescue . . .

PORTICO



1		Header and PDF, 100 articles, 200 files				Full Text, 94 articles, 1045 files			
2	Activity	Processing time (secs)	Persistence time (secs)	Total Activity Time	% of total	Processing time (secs)	Persistence time (secs)	Total Activity Time	% of total
3	Initialize	13.40	4.09	17.49	0.40	0.65	3.88	4.53	0.04
4	Virus Scan	266.30	1.12	267.43	6.05	209.78	1.18	210.96	1.72
5	Exclusion Rules	0.34	0.13	0.48	0.01	4.91	0.07	4.99	0.04
6	Verify Checksum	0.33	0.14	0.46	0.01	0.33	0.12	0.44	0.00
7	Override Rules	0.31	0.07	0.38	0.01	0.28	0.08	0.36	0.00
8	Remove Layer	8.02	217.76	225.78	5.11	48.67	1,301.02	1,349.69	10.97
9	Virus Scan	376.04	73.83	449.87	10.18	397.35	376.63	773.98	6.29
10	Exclusion Rules	26.29	0.14	26.43	0.60	86.35	0.31	86.66	0.70
11	Verify Checksum	5.38	0.11	5.49	0.12	27.94	0.23	28.17	0.23
12	Override Rules	8.41	5.85	14.26	0.32	28.89	5.26	34.14	0.28
13	Remove Layer	5.38	0.14	5.52	0.13	25.73	0.24	25.97	0.21
14	Unit Identity	12.17	400.53	412.70	9.34	91.63	1,088.33	1,179.96	9.59
15	Completeness Check	32.73	0.18	32.91	0.75	115.55	0.40	115.94	0.94
16	Verify Metadata File	114.71	45.31	160.02	3.62	134.49	38.18	172.67	1.40
17	Transform SGML/XML	183.34	183.19	366.53	8.30	328.88	205.95	534.83	4.35
18	Verify Transformed File	102.68	46.96	149.64	3.39	124.13	47.62	171.75	1.40
19	Extract File References	42.34	7.38	49.72	1.13	69.97	201.33	271.30	2.21
20	Resolve File References	23.01	0.14	23.15	0.52	140.84	103.91	244.74	1.99
21	Verify Other Files	53.00	43.25	96.25	2.18	257.13	429.75	686.88	5.58
22	Generate DMD	67.86	93.12	160.98	3.64	107.79	90.51	198.30	1.61
23	Curate DMD	64.54	99.59	164.13	3.72	76.10	99.22	175.32	1.43
24	Replace File References	4.65	5.34	9.99	0.23	107.19	166.79	273.99	2.23
25	Verify Article File	8.39	0.16	8.55	0.19	72.52	28.57	101.09	0.82
26	Generate all TMD	239.59	253.05	492.64	11.15	624.24	1,088.52	1,712.76	13.93
27	Completeness Check	35.65	21.84	57.49	1.30	108.02	86.05	194.07	1.58
28	QC Messages	100.25	0.20	100.46	2.27	166.53	0.47	167.00	1.36
29	Sampling	1.57	5.51	7.08	0.16	1.52	4.67	6.19	0.05
30	Generate Checksum	68.36	121.20	189.56	4.29	327.23	524.02	851.25	6.92
31	Generate METS	297.46	558.49	855.95	19.38	1,009.63	1,614.86	2,624.49	21.34
32	Verify METS	65.58	0.31	65.89	1.49	96.42	0.12	96.54	0.78
33		2,228.10	2,189.11	4,417.21		4,790.65	7,508.29	12,298.94	

Planning the Plan

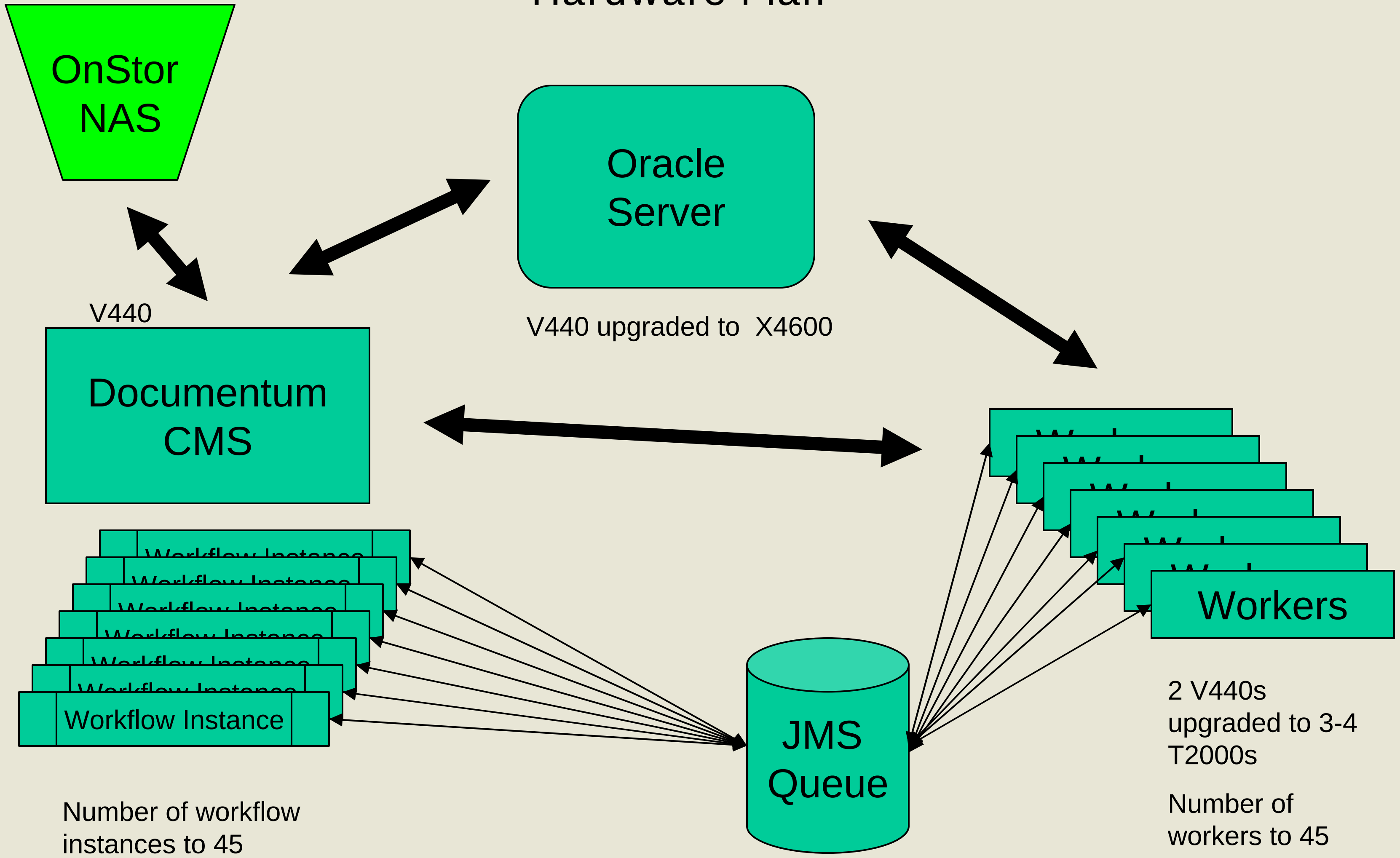
	Test Batch		Elapsed Time		Planned Improvement		Resulting Time per Article
	# Articles	# Files	Batch	Article	Persistence	Processing	
Header	100	200	4,417	**44.17**	45%	20%	**15.46**
Full Text	94	1045	12,298	**130.80**	50%	20%	**39.25**

GOALS AND ASSUMPTIONS FOR 2007	
Ingested Articles	6,000,000 / year
% Rerun	30%
% Updates	3%
Total Capacity	8,000,000 / year
% Full Text	50%
Processing time	218,833,745 secs

Software Plan

- Prioritized by bang for buck
- Rewrite of data persistence
  - 45-50% performance improvement and allows further hardware scaling
  - Replace Documentum attributes / folders with Oracle tables
    - Original design abstracted Documentum at cost of performance
    - New design is idiomatic Documentum application architecture
  - Major shift in system resource demands from Documentum to direct Oracle connections
  - Required major rewrite of user interface data access layer
- Simplify system design
  - Eliminate distributed tool processing in favor of embedded mode
    - Move all tools into memory space of worker
    - Eliminated 50% of JMS messages
    - No actual business use case for this element of design
  - Eliminate trivial use of web services
- Other ideas considered (not all implemented yet)
  - Streamline workflow activities
  - Eliminate duplicate JHOVE call
  - Eliminate use of XML blobs prior to METS generation

Hardware Plan



PART III
“Putting it together, bit by bit”

Workers on T2000s

- T2000s were new machines at that time
  - Risk to go with new machine
  - Performed very well
- In abstract testing number of threads running should exceed number of hardware threads
  - E.g., 32 hardware threads can support 40+ running threads
- In practice our application can only use about 15 threads
  - Variable memory requirements for XML parsing, JHOVE, etc.
    - Must leave room for traffic jams, as it were
    - Limitation of JVM in dealing with out of memory errors
  - Limited by Documentum 32bit libraries to 4GB of memory
  - Limited by forking problems in Sun JVM

Oracle on X4600

- X4600 was ideal for Oracle
  - Oracle load went from 90% to 50-60% while executions went up 20 fold
  - Significant improvement over V440
- Still have to tune for your particular application
  - Took 6 weeks of DBA and QA staff time and lots of testing
- Ramped from 250 executions / second to 5,000 / second
  - System capacity went from 1200/header articles per hour to 2400, 3600, 5000, 6300, then 7000 in a matter of two weeks
- Default statistics algorithms fail for large databases
  - Oracle reverted to full-table scans bringing system to a crawl
  - Solution was to force frequent statistics collection
- Other optimizations
  - Always using prepared statements
  - Adjust cache size for LOBs
  - Separate LOB area from regular table space

Documentum

- Documentum performed well at large numbers of objects
  - Repository scaled to 100M without any hiccups
- V440 was adequate; no upgrade needed
- Limitation was performance on adds and deletes
  - Documentum is human-scaled
  - Adds are slow
  - Deletes are very slow
    - We rewrote the deletion routines
  - We add and delete up to 1M objects per day

Tuning the Entire System

- Balancing all the pieces
  - # of Workflow instances in Documentum
  - # of Workers
    - # of T2000s
    - # of JVMs on each T2000
    - # of Worker threads in each JVM
- Many combinations to test
- Performance peaked and then declined when too many threads were added
  - Cause never fully understood: network, oracle, JVM?
  - Fortunately we had well exceeded our performance goals by that point

Testing at Scale

- How to execute very large tests?
  - Staggered start is important
  - Mixed content sizes and types is essential
  - Create random traffic effects
  - Test both good and bad data
- Test on production equipment
  - Not a smaller installation
  - No substitute for large quantities on big machines
- Can you afford to run a month-long test?
  - Can you afford not to?
- We got caught
  - Tested > 1M articles
  - But not in a continuous run
  - Found additional problems once system went live and ran for extended period without rebooting

When to Declare Victory?

- A critical question
- How much tuning is enough?
  - Cost benefit analysis based on unknown costs
- Hard to walk away from great ideas for further improvements
  - Defer to later releases

PORTICO

PART IV
The Challenges of Scale

System Impact of Scale

- Other Pieces of the System
  - All the pieces of the system must be scaled up
  - On ramps & off-ramps, not just the highway
  - We had to rewrite our scripts, loaders, all the pieces around the edges of the system
  - More than first expected
- System Admin
  - New cleanup and maintenance jobs were necessary
  - New deletion and purging routines
  - New logging and log management
- User Interface
  - Usability: scale may well prompt redesign
  - Raw performance
- Storage / Backup / Recovery
  - Differentiation becomes desirable, cost effective, and even essential
  - Different types of disk for work areas, holding areas, archive
  - Different backup strategies for each part of the system
    - You know you are not in Kansas anymore when a backup takes 157 hours!

Human Impact of Scale

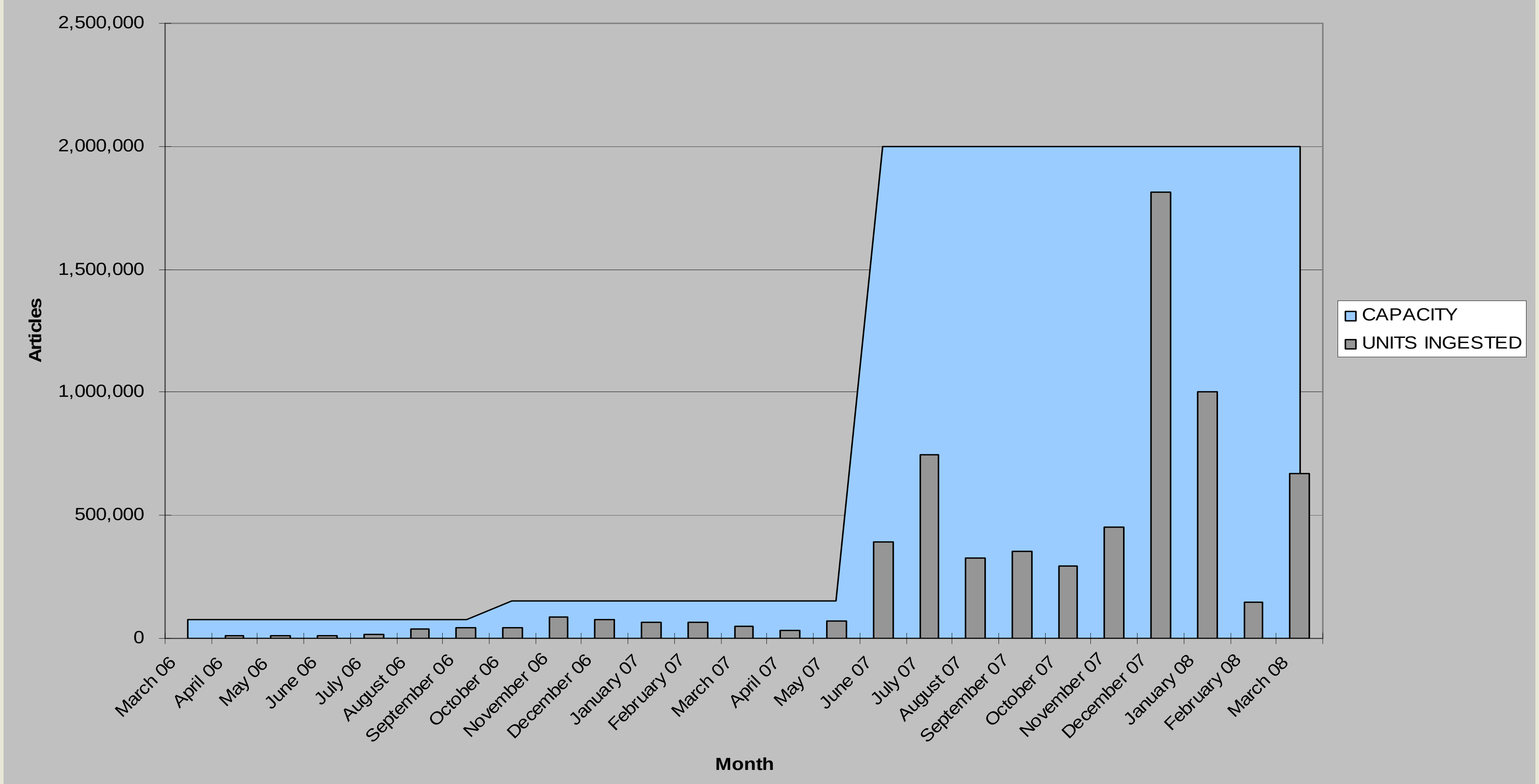
- Human roles in our system
  - Loading of content
  - Quality Assurance for new content streams
  - Problem Resolution during processing
- Scale changes everything
  - Human tasks don’t scale gracefully
  - Can’t look at 1,000 objects and see a pattern
  - Can’t push a button 1,000 or 100,000 times
  - Problem solving becomes much more technical
- New User Interface Requirements
  - Reports, summary view, classifications
  - Facilities to execute actions against sets of objects
    - “My Batches”
- Hard for users to imagine the impact of scale
  - Particularly for people used to the existing system
  - Easier with an entirely new system
  - Bring in UI consultants

Organizational Impacts of Scale

- It isn’t easy to keep a dragon well fed!
- Lead time in getting content organized and ready for ingest
- Challenges of mixing small and large content streams
  - Can work for 2 months on 10K articles
  - Or a week on 200K articles
- Note to charts on next two slides:
  - Assumption is system would run at an average 50-75% of peak capacity, not 100%

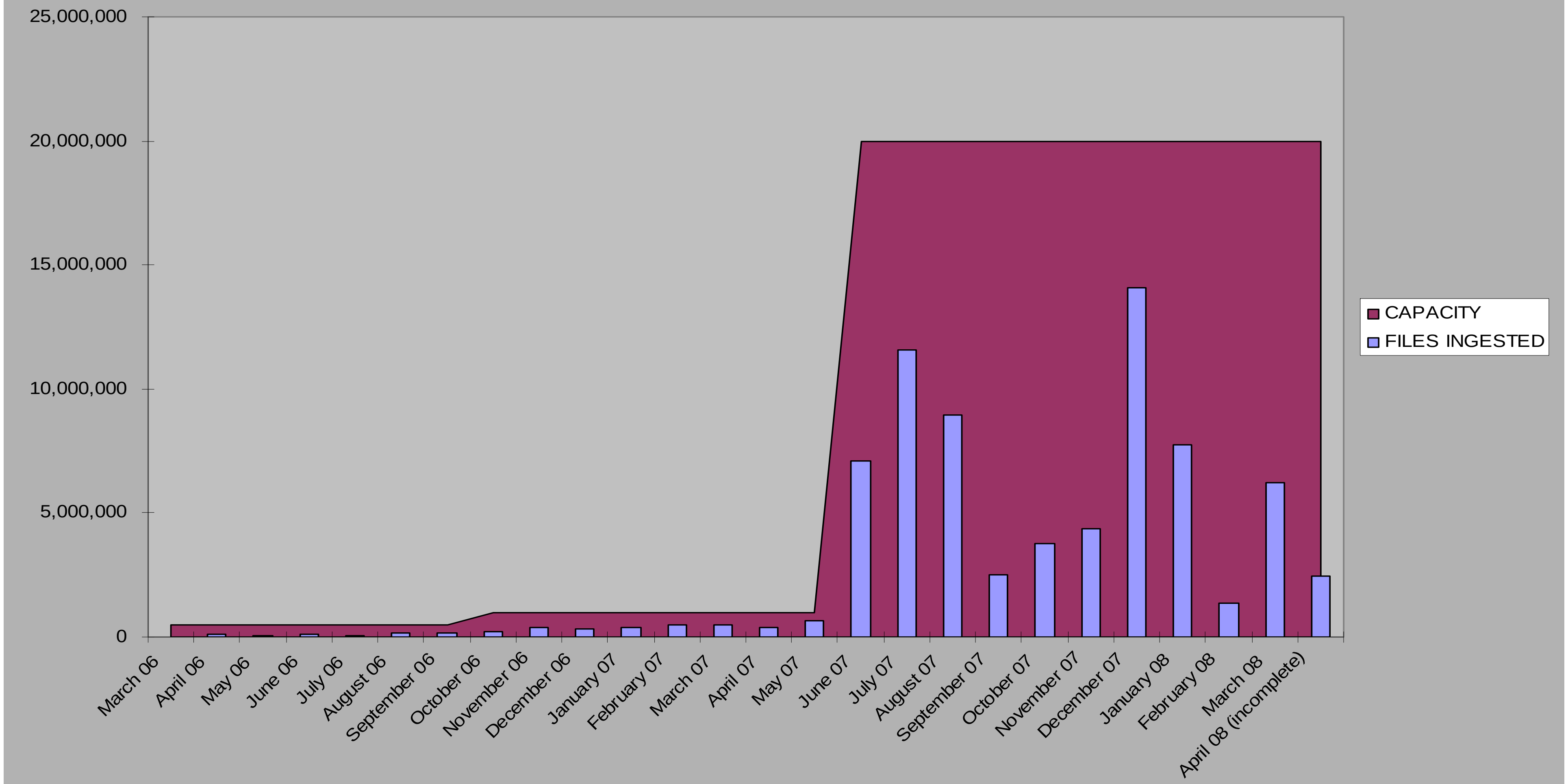
PORTICO

Monthly Article Ingest versus Capacity



PORTICO

Month File Ingest Versus Theoretical Capacity



PORTICO

Some Thoughts about Scale

- Scale affects everything before, after, and around the system
- Nothing is simple or quick in units of millions
- Batch processing redux:
  - Schedule, wait, evaluate results, schedule again
  - PC mode versus old mainframe batch processing mode
  - What % of idle capacity is acceptable?
- High-capacity systems require large amounts of testing and big tests
- Allow lots of time for integration & tuning; it pays off
- Make sure you know how you are going to feed your dragon
  - All parts of the organization have to on board with scaling up
  - Technology alone cannot solve the problem
- Expect surprises!