Accelerating DITA with OmniMark

A Scalable Solution for Demanding Production Environments

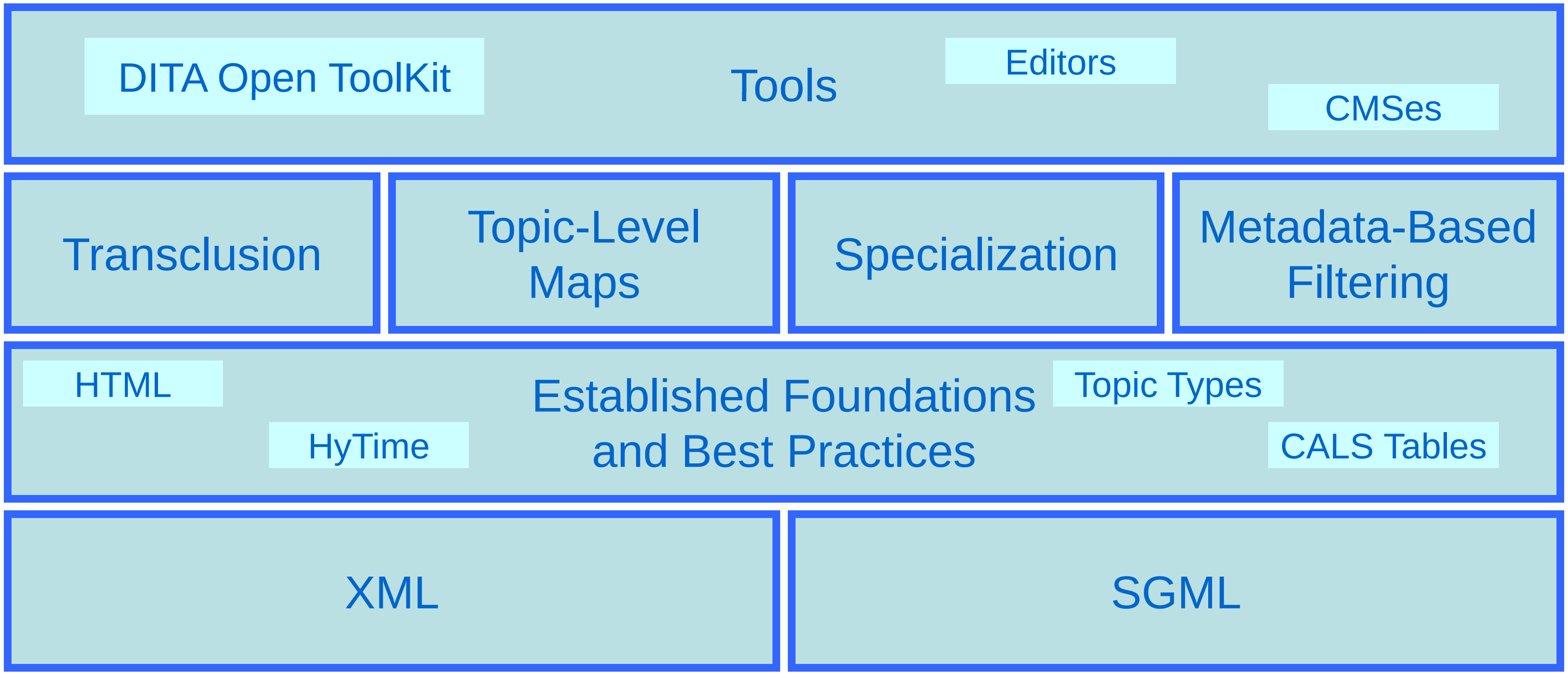
XML-in-Practice 2008

ramodeo@stilo.com

Darwin Information Typing Architecture (DITA)

- An OASIS standard for content

- Reduce
- Reuse
- Repurpose



Tools

DITA Open ToolKit

Editors

CMSes

Transclusion

Topic-Level Maps

Specialization

Metadata-Based Filtering

Established Foundations and Best Practices

HTML

HyTime

Topic Types

CALS Tables

XML

SGML

- DITA Publishing
  - Depends on efficient assembly, interpretation, filtering & formatting of content components

The DITA Open Toolkit Factor

- The Open Toolkit has been a big part of DITA's success
  - Open source
  - Active development community
  - Thorough implementation of DITA
  - Out-of-the-box support for multiple output formats
  - Modular architecture
  - Easily customized
- Components of the Open Toolkit are replaceable
  - Users have a choice of XSLT and FO processor components
- Many commercial products bundle the Open Toolkit
- As a result DITA is closely identified with the Open Toolkit

DITA Editors incorporating Open Toolkit

Adobe Framemaker 8	✓
Information Mapping Content Mapper	✓
Inmedius DITA Storm	✗
In.vision DITA Studio	✗
Justsystems XMetaL Author Enterprise 5.1	✓
PTC Arbortext 5.3	✗
SyncRO Soft <oXygen/> 9.1	✓
Syntext Serna 3.5	✓
XMLmind XML Editor 3.6	✓

Source: STC intercom, April 2008 (DITA issue)
DITA Tools from a to z,
Bob Doyle
http://www.ditanews.com/tools/STC_Intercom.pdf

DITA CMS Integration with Open Toolkit

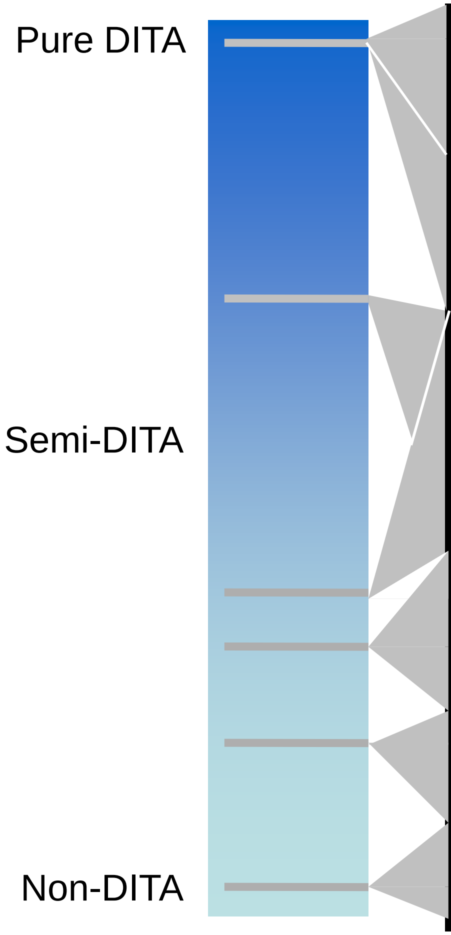
Astoria On Demand	✓
Author-it	✗
Bluestream XDocs	✓
DITA Exchange	✓
DocZone	✗
Inmedius Horizon	✗
IXIASOFT DITA CMS Framework	✓
PTC Arbortext Content Manager	✗
SiberLogic SiberSafe	✓
Trisoft Infoshare	✓
Vasont	✓
X-Hive Docato	✓
XyEnterprise Content@	✓

Source: STC intercom, April 2008 (DITA issue)
DITA Tools from a to z,
Bob Doyle
http://www.ditanews.com/tools/STC_Intercom.pdf

Exploiting DITA

- As DITA evolves, it will be applied to ever more demanding situations
  - Many industries publish huge volumes of data
    - Aerospace, automotive, oil services, legal publishing
- Aspects of DITA can be used for their own sake
  - DITA specialization may spin off into its own standard
  - Transclusion can allow reuse even among monolithic documents
  - Metadata-based filtering can provide general-purpose effectivity support
  - DITA is a very modular specification
- Some of these scenarios will have very demanding requirements
  - Very large "topics"
  - Large numbers of topics

The DITA Continuum at Stilo



Content	Details	Drivers	Topic-based	Specialization	Filtering	Transclusion	Uses OTK
Semi-conductor Datasheets	FrameMaker source; PDF publishing	Authoring costs; Consistency; Customized Pubs	●		●	●	●
Legal Procedures	e-Learning; Word and HTML source	Adaptable; Simplified authoring; Integration with existing XML tools	●	●	●	●	●
Aerospace Standards, 2 projects	Monolithic; SGML, Interleaf, Word source; publish to ATA, S1000D, new web services	Many legacy formats; Multi-target; access to sub-contractors; S1000D support	◐	●	●	●	◐
Aircraft Maint. Manuals	Monolithic; ATA source; E-manuals	Efficient update; Targeting; Costs; Regulatory compliance	◐	◐	◐		
Automotive	Monolithic; Multiple sources; SGML	Efficient update; Targeting; Costs;	◐	◐	◐		
Software Docs	Topics; SGML; RDBMS storage	Authoring costs; Multi-target; Reuse;	●	◐		◐	

Pushing the Boundaries

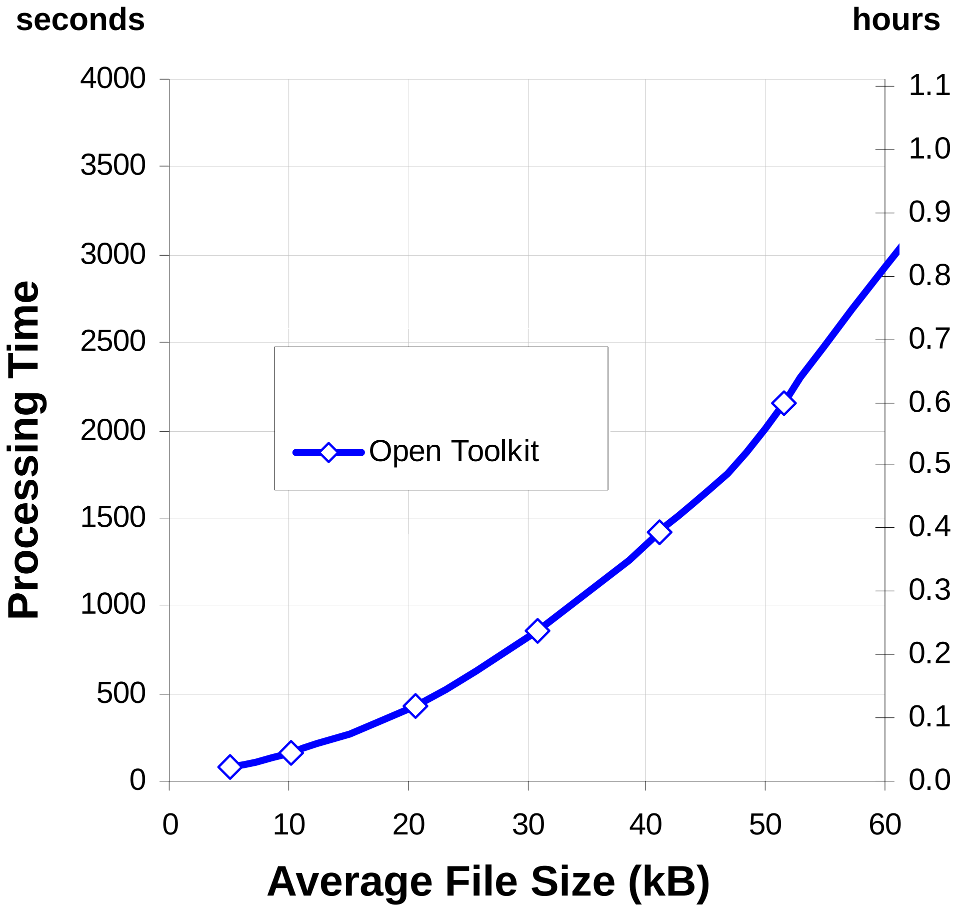
- How well does the Toolkit cope with these situations?
- The Toolkit has a modular architecture
  - It can be used as a base for partial DITA applications
- Some coding tricks are required
  - XSLT rules must be implemented carefully to preserve support for specialization
- Most importantly, XSLT is not known as a fast processing technology
  - Can the Toolkit cope with high volumes of data?
- We can test this

Building a DITA Stress Test

- Sample input is the DITA language reference
  - 200+ topics
  - 1468 conref references
  - 741 targets referenced by conref
  - 1.06 MB
  - Average file size 5 kB
- The DITA language reference was inflated in two ways
  - Topic sizes were increased up to a factor of 100 (to 500KB per file)
  - Number of files was increased up to a factor of 100 (to 20,000 files)
- To increase topic sizes
  - The body of each topic was replicated
  - A random prefix was added to each word to create unique content
  - The number of links increased proportionately
- To increase the number of files
  - The whole topic was replicated
  - A random prefix was added to each word, each id, and each idref
  - The number of links and link targets increased proportionately

Open Toolkit performance (1)

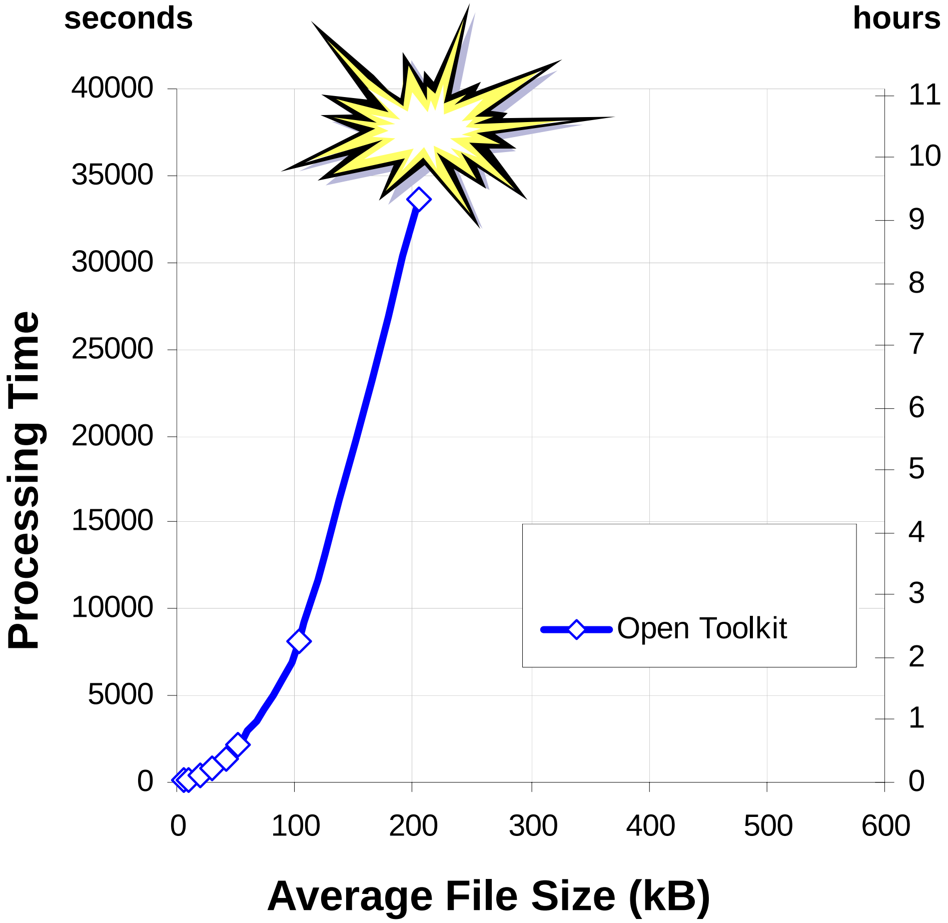
Processing Time vs. Average File Size: from 5 kB to 50 kB



AVG SIZE (kB)	Open Toolkit TIME (s)
5	80
10	156
21	428
31	852
41	1422
51	2160

Open Toolkit performance (2)

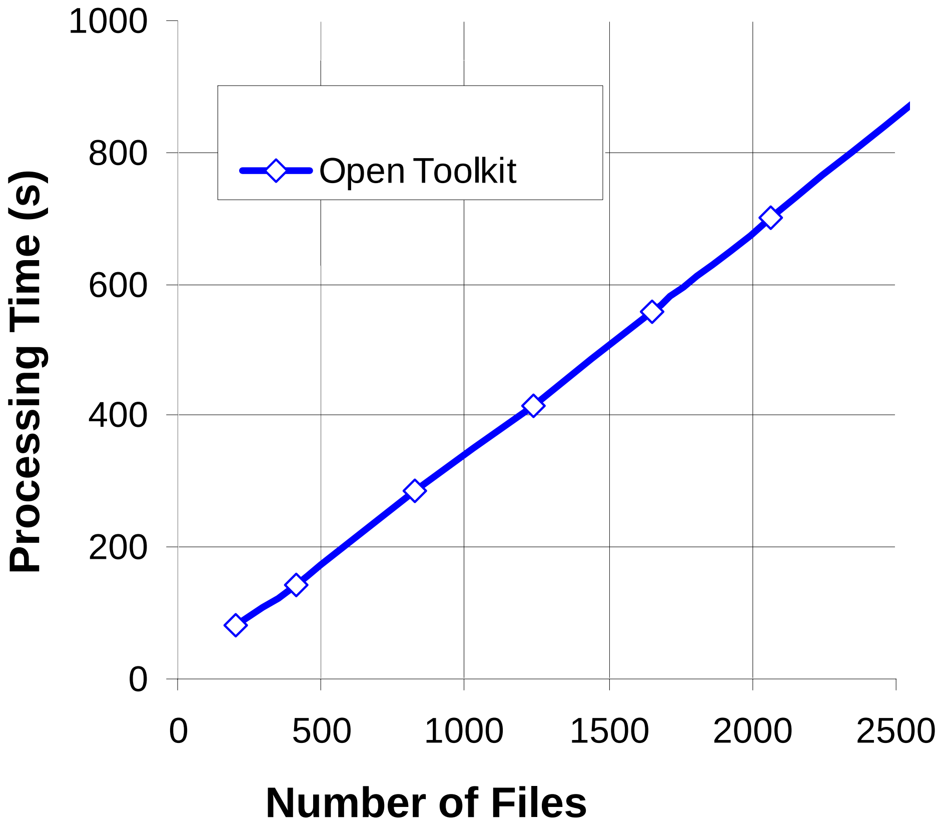
Processing Time vs. Average File Size: from 5 kB to 500 kB



AVG SIZE (kB)	Open Toolkit TIME (s)	DITA Open Toolkit TIME (hr)
5	80	0.02
10	156	0.04
21	428	0.12
31	852	0.24
41	1422	0.40
51	2160	0.60
103	8160	2.27
206	33660	9.35
309	***OUT OF MEMORY***	
412		
515		

Open Toolkit performance (3)

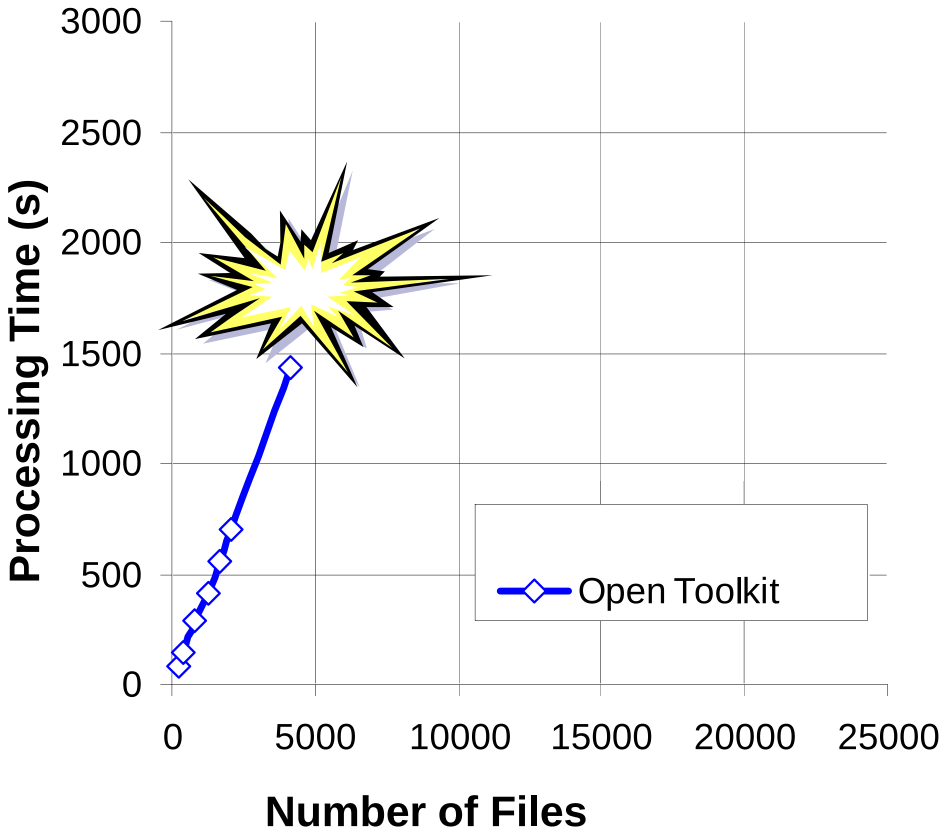
Processing Time vs. Number of Files: from 200 to 2,000



Number of Files	DITA Open Toolkit TIME (s)
206	80
412	144
824	286
1236	415
1648	557
2060	699

Open Toolkit performance (4)

Processing Time vs. Number of Files: from 200 to 20,000



Number of Files	Open Toolkit TIME (s)
206	80
412	144
824	286
1236	415
1648	557
2060	699
4120	1429
8240	***OUT OF MEMORY***
12360	
16480	
20600	

Accelerating DITA for Production

- An alternative to the Toolkit is required
- Production-level quality
  - No limits on large volumes of content
  - Consistently high throughput speed as volume increases
  - Robust and maintainable
- Rapid development architecture
  - Out-of-the-box rendering for standard DITA schemas/DTDs
  - Easily customized
- DITA-aware
  - Built-in support for DITA concepts
    - Transclusion
    - Specialization
    - Filtering
  - No programming tricks required

OmniMark DITA Accelerator

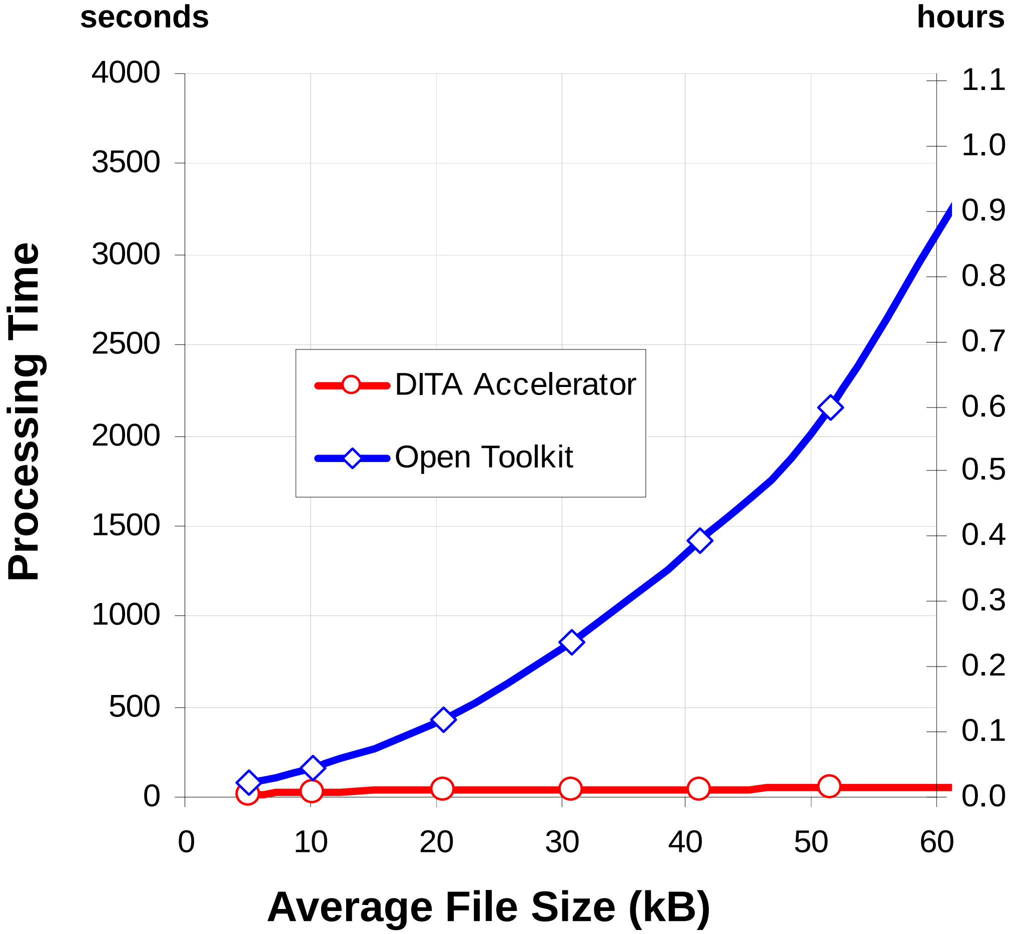
- DITA Accelerator implements HTML publishing
  - Implements all functionality required for language reference
  - HTML support still requires completion
  - PDF to be implemented in the future
- Behavior is modeled on the Toolkit
  - Automated tests were written to ensure that the output is almost identical
  - The output of the DITA Accelerator is nearly identical to the Open Toolkit
    - index.html from the Open Toolkit
    - index.html from the DITA Accelerator
  - Some small differences remain
    - Table cell borders are inconsistent in some cases
    - Some errors in the DITA toolkit are corrected in the Accelerator
- High performance is achieved with streaming technology
  - Leverages OmniMark's built-in support for streaming
  - Makes heavy use of referents
- A DITA-aware library has been implemented
  - Programmers do not have to employ coding tricks

Gentlemen, start your engines

- DITA language reference
  - 206 files
  - 1414 elements with ids (potential link or conref targets)
  - 1468 conref references
  - 741 targets referenced by conref
  - 1.06 MB
  - Average file size 5 kB
- Initial results are promising
  - DITA Open Toolkit: 1 minute, 21 seconds
  - DITA Accelerator: 18 seconds
  - Speed improvement: 4X
- What about larger input sets?

Comparing DITA Accelerator and Open Toolkit (1)

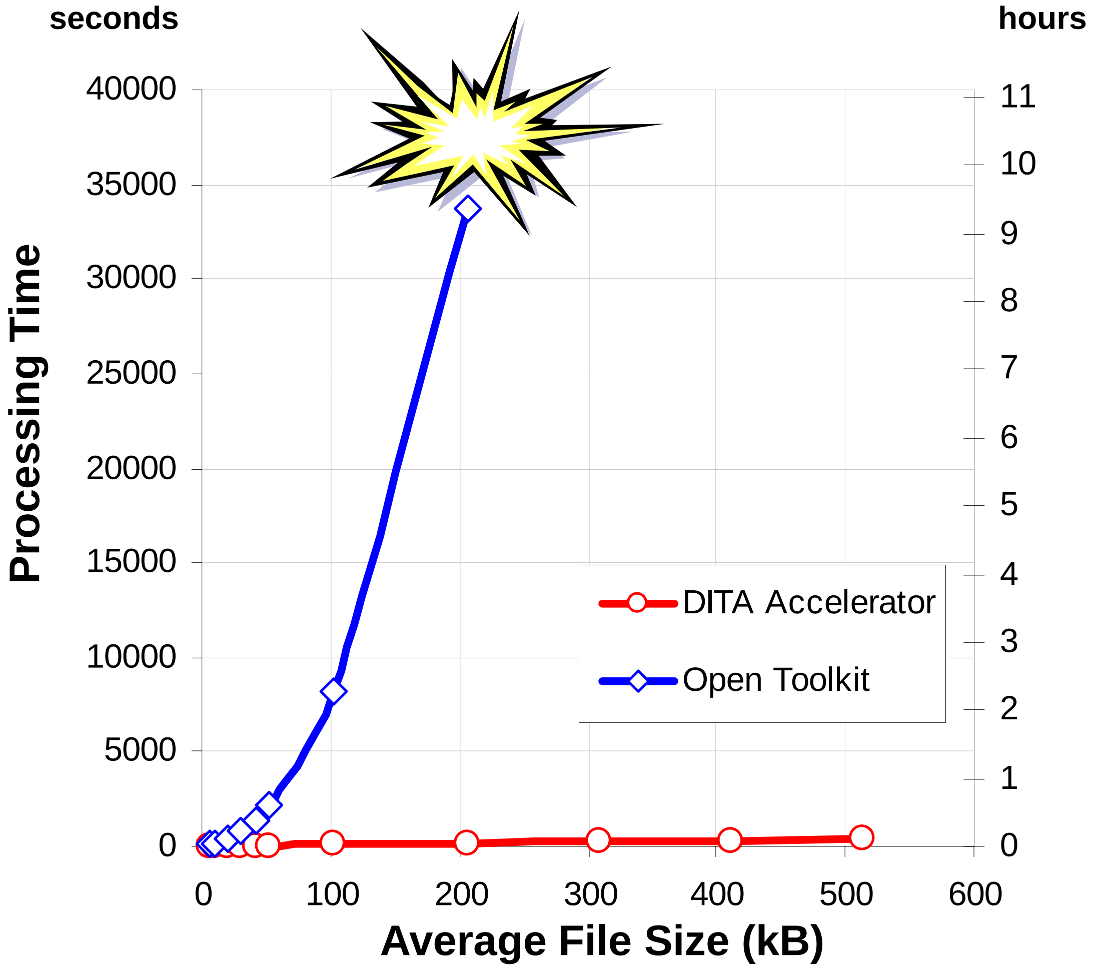
Processing Time vs. Average File Size: from 5 kB to 50 kB



AVG SIZE (kB)	Open Toolkit TIME (s)	DITA Accelerator TIME (s)
5	80	18
10	156	20
21	428	35
31	852	41
41	1422	46
51	2160	57

Comparing DITA Accelerator and Open Toolkit (2)

Processing Time vs. Average File Size: from 5 kB to 500 kB



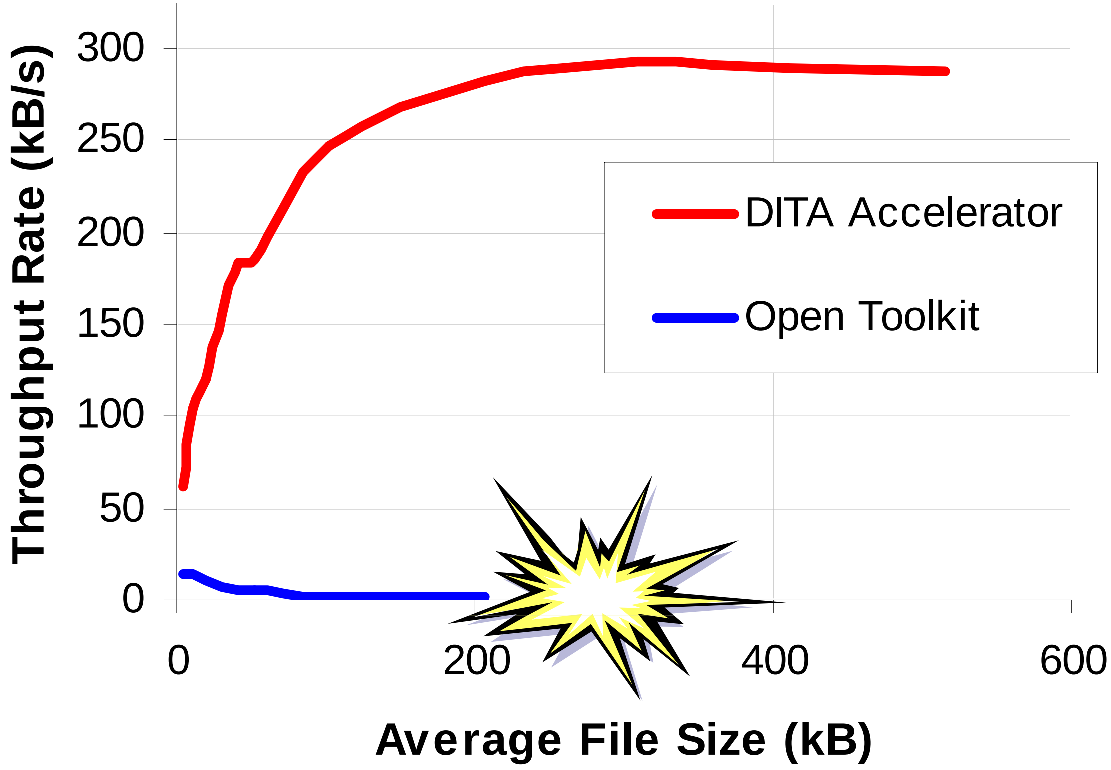
AVG SIZE (kB)	Open Toolkit TIME (s)	DITA Accelerator TIME (s)
5	80	18
10	156	20
21	428	35
31	852	41
41	1422	46
51	2160	57
103	8160	86
206	33660	150
309	*OUT OF MEMORY*	217
412		292
515		369

= 9 hours

= 6 minutes

Comparing throughput rate as sizes increase

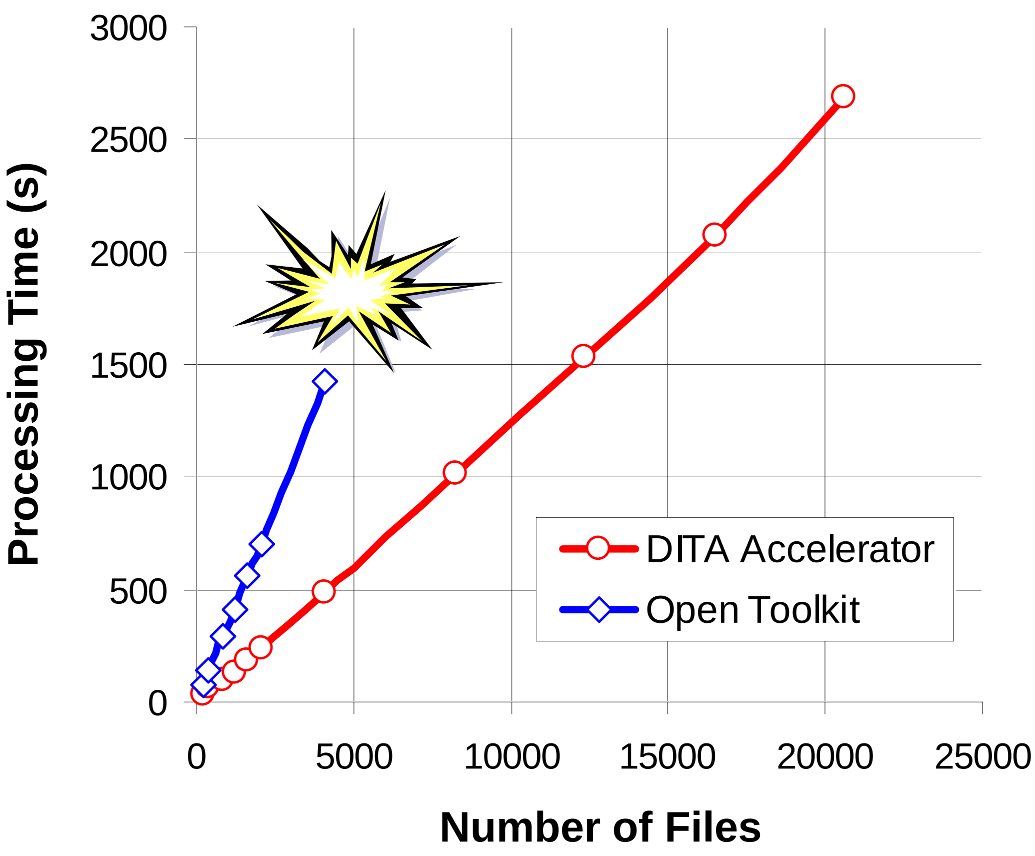
Processing Throughput as File Size Increases



AVG SIZE (kB)	Open Toolkit THROUGHPUT (kB/s)	DITA Accelerator THROUGHPUT (kB/s)
5	13.3	62
10	13,6	104
21	9.9	120
31	7.5	155
41	6.0	184
51	4,9	186
103	2.6	247
206	1.3	283
309	*OUT OF MEMORY*	293
412		290
515		287

Comparing DITA Accelerator and Open Toolkit (3)

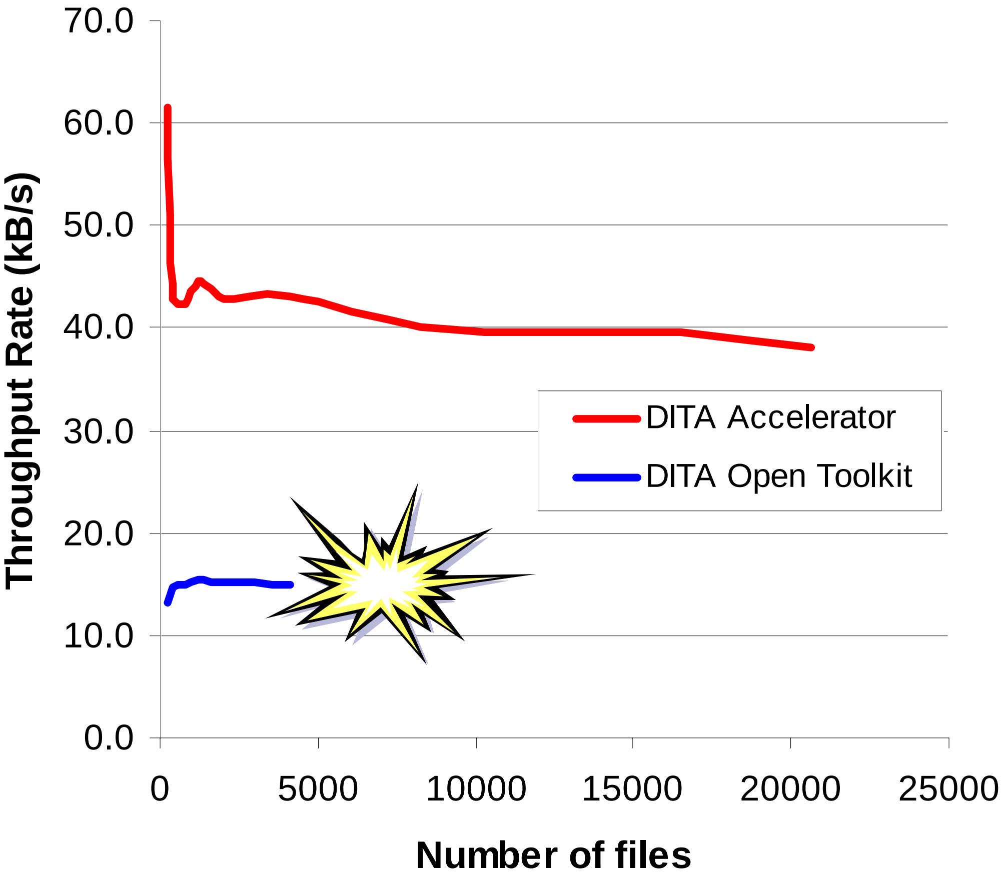
Processing Time vs. Number of Files: from 200 to 20,000



Number of Files	Open Toolkit TIME (s)	DITA Accelerator TIME (s)
206	80	18
412	144	49.5
824	286	100.2
1236	415	142.4
1648	557	193.8
2060	699	247.1
4120	1429	491.2
8240	***OUT OF MEMORY***	1055.2
12360		1601.5
16480		2143.4
20600		2788.5

Throughput rate as number of files increases

Number of Files	Open Toolkit THROUGHPUT (kB/s)	DITA Accelerator THROUGHPUT (kB/s)
206	13.3	35
412	14.7	35
824	14.8	42
1236	15.3	47
1648	15.2	47
2060	15.2	45
4120	14.8	43
8240	***OUT OF MEMORY***	42
12360		42
16480		41
20600		39

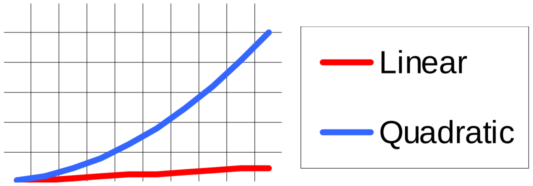


Interpretation of timing statistics

- DITA Open Toolkit is best for light duty
  - Performance degrades rapidly as file sizes increase
  - Performance is fairly flat as the number of files increase
  - In both sets of tests, the toolkit eventually fails when it runs out of memory
  - A great starting point
- OmniMark DITA Accelerator is robust and scales well
  - Does not run out of memory
  - Throughput rate is fairly flat in both types of testing
- DITA can play in demanding production environments
  - Because DITA is a standard, technology can be changed without changing the information architecture

Ongoing analysis

- Tests used DITA Toolkit "out-of-the-box"
  - Different XSLT processors may improve performance
- Forum discussions suggest a workaround for *memory exhaustion*
  - Reload XSLT stylesheet on every transformation
  - Currently requires toolkit modification (may be configurable in 1.5)
  - Expect slower performance on smaller topics
- Even with improvements, best performance will still be quadratic for increasing file sizes



- There will be room for improvement for the foreseeable future

Role of OmniMark

- Most of the performance is due to engineering "behind the scenes"
  - Native efficiency of OmniMark
  - Streaming architecture reduces memory requirements
  - Record shelves can be used to implement high speed lookup for DITA processing rules
- OmniMark referents simplify support for transclusion
  - Referents are a streaming mechanism for reordering content
  - Eliminate complex book-keeping
- OmniMark language is easily extended
  - Macros
  - Modules (functions and data types)
- Bonus: SGML support included

Usability

- XSLT supports DITA reluctantly
- XSLT rule selection mechanism is not DITA-aware
  - Two templates that match the element "u":

    ```
    <xsl:template match="*[contains(@class,' hi-d/u ')]">
    <xsl:template match="*[contains(@class,' topic/ph ')]">
    ```
  - Both have equal priority
    - Programmer must use tricks to ensure that the "hi-d/u" takes precedence over the "topic/ph" rule
    - Extra conditions on the "topic/ph" rule can invert the hierarchy!
- The spaces around the class names are required
  - And no more than one on each side
  - XSLT does not enforce this
  - Programmer must code carefully to avoid inexplicable behavior

OmniMark extensions provide DITA support

- DITA Accelerator augments OmniMark with "DITA rules"
  - Automatically prioritized according to the specialization hierarchy
  - Rule selection is optimized so that performance stays consistent as more rules are added
  - DITA rules can be grouped into sets, like OmniMark rules
  - DITA rules can be supplied as OmniMark modules
  - Local DITA rules take precedence over imported rules for the same DITA class
- Module supplies support functions that understand DITA class specialization

DITA Accelerator specialization support

- The syntax of DITA rules is based on OmniMark element rules
  - Element rules specify element names

```
element "u"
   output "<u>" || "%c" || "</u>"
```

  - DITA rules specify classes instead of elements

```
declare dita-rule hi-d-u-rule
        class     "hi-d/u"
   output "<u>" || dita.process-content || "</u>"
```

  - DITA rule for "hi-d/u" will take precedence over "topic/ph"
    - Based on the class specialization in the DTD
- Currently implemented by macros
  - Allows access to full OmniMark language
- DITA module also provides utility functions
  - DITA class-based queries for current and ancestor elements
  - Mimics the element tests built into OmniMark

Selection by DITA class – understands specialization

Processes content, like "%c" in element rules

Conclusions

- The OmniMark-based DITA Accelerator provides scalability
  - Robust
  - Consistent throughput as volumes increase
  - No catastrophic failures
- The OmniMark language can be easily extended to provide a natural DITA programming environment
  - Programmers can "think in DITA", rather than trying to align a pre-existing programming model with the DITA semantics
- Standards are about choice of tools
  - DITA Toolkit is a good choice for
    - Learning DITA
    - Prototyping
    - Less demanding production uses
  - OmniMark DITA Accelerator
    - Demanding production environments
- Most importantly, tool choice must be governed by the unique characteristics of your environment