Profiling OGSA-DAI Performance for Common Use Patterns

Profiling OGSA-DAI Performance for Common Use Patterns

UK e-Science

All Hands Meeting 2006

Bartosz Dobrzelecki

EPCC, The University of Edinburgh

OGSA-DAI

- Web Services interface to databases
- An *extensible framework* for data access and integration
- Expose heterogeneous data resources to a grid through web services
  - Relational
  - XML
  - File based
  - User provided (extensibility point)
- Interact with data resources
  - Queries and updates
  - Data transformation / compression
  - Data delivery
  - Application-specific functionality
- A base for higher-level services
  - Federation, mining, visualisation,…

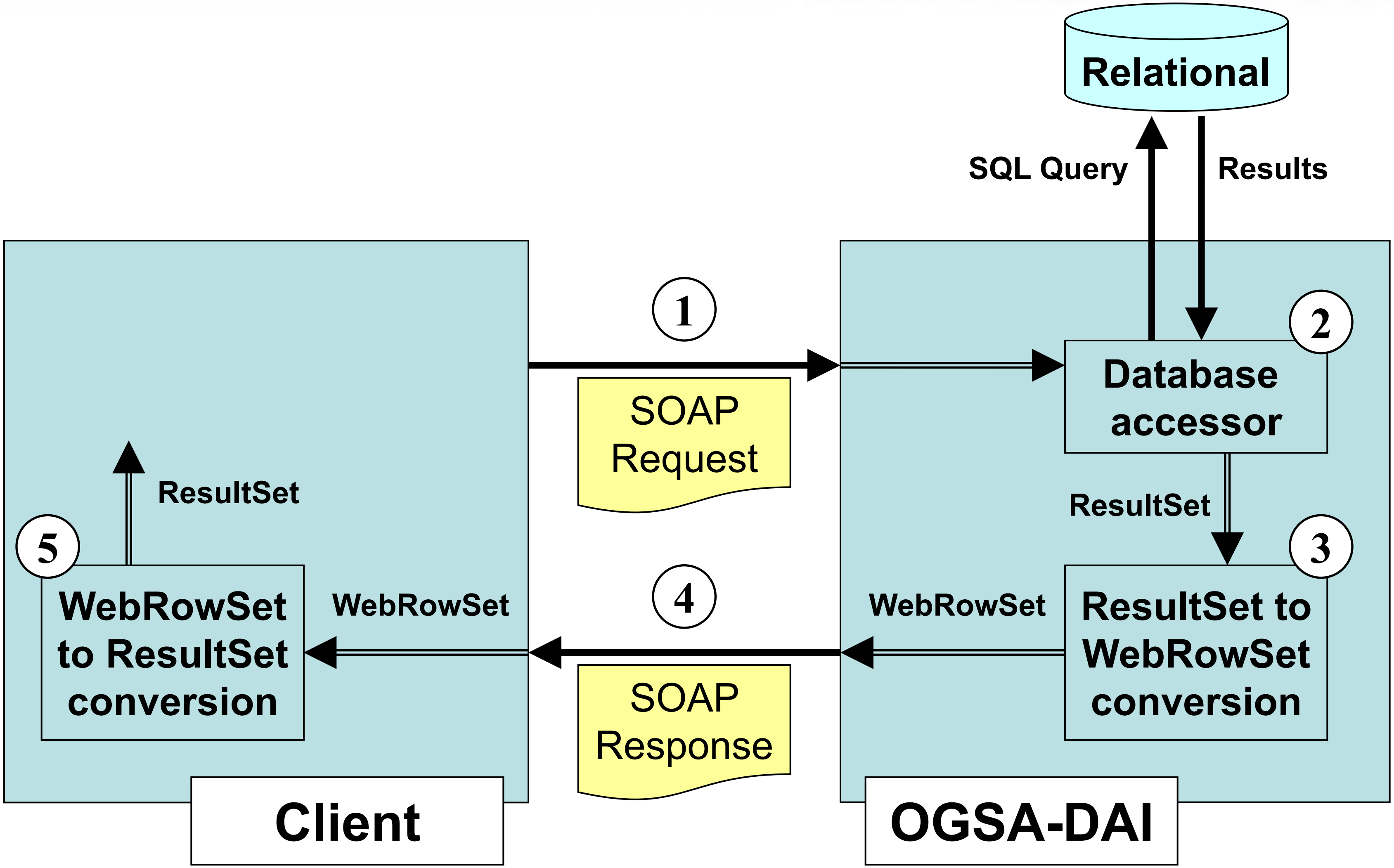
Common usage patterns

- Have selected two typical use patterns
  - Use these as a basis for improving the performance

- First use pattern: SQL query
  - Client runs an SQL query on a remote OGSA-DAI service
  - OGSA-DAI service returns the query results to the client
  - Results are contained in an XML document

- Second use pattern: User accesses binary data
  - Binary data could be files or BLOBs in a database
  - Data is exposed by an OGSA-DAI service
  - Encoded data is delivered to a client in an XML document

First use pattern: Executing an SQL Query



Improvement 1: Faster Conversion

Bottleneck:

- Conversion between ResultSet (object) and WebRowSet (XML)
  - Large number of String to bytes conversions

Improvements:

- Restricted conversion framework to text based formats only
  - Data represented internally as char sequence

- Improved the performance of XML production
  - To produce valid documents special XML characters need to be escaped
  - Previously used regular expressions Java API to do this
  - For large number of rows this process becomes very expensive
  - Have implemented a much more efficient parser to perform this task

Improvement 2: Change in Data Format

Bottleneck

- WebRowSet format is only used for intermediate delivery
  - Adds significant amount of mark-up to describe data
  - More data hence it affects message transfer times
  - XML is still expensive to parse

Improvement

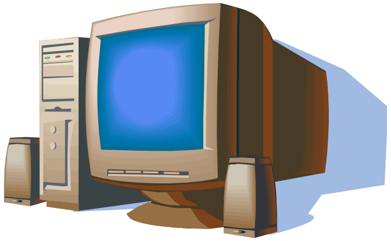
- Instead use CSV (Comma Separated Values) as an alternative
  - More lightweight
  - Easier to parse document format
- For example to represent one row:

Drawbacks

- No metadata (optional line with column names)
  - Could be delivered in separate stream as WebRowSet metadata
- CSV is not standardised - used consistently within OGSA-DAI

Experimental Setup

- Container
  - –Apache Tomcat 5.0.28
- Globus
  - –Globus Toolkit WS-Core 4.0.1
- OGSA-DAI
  - –OGSA-DAI WSRF v2.1
  - –OGSA-DAI WSRF v2.2
- Machines
  - –Server
    - –Sun Fire V240 with dual 1.5GHz UltraSPARC IIIi and 8GB RAM
    - –Solaris 10 and J2SE 1.4.2_05
  - –Client
    - –Dual 2.4GHz Intel Xeon system with
    - –RedHat 9 Linux and J2SE 1.4.2_08



Experimental setup (cont.)

JVM flags

- `-server -Xms256m -Xmx256m`

Network

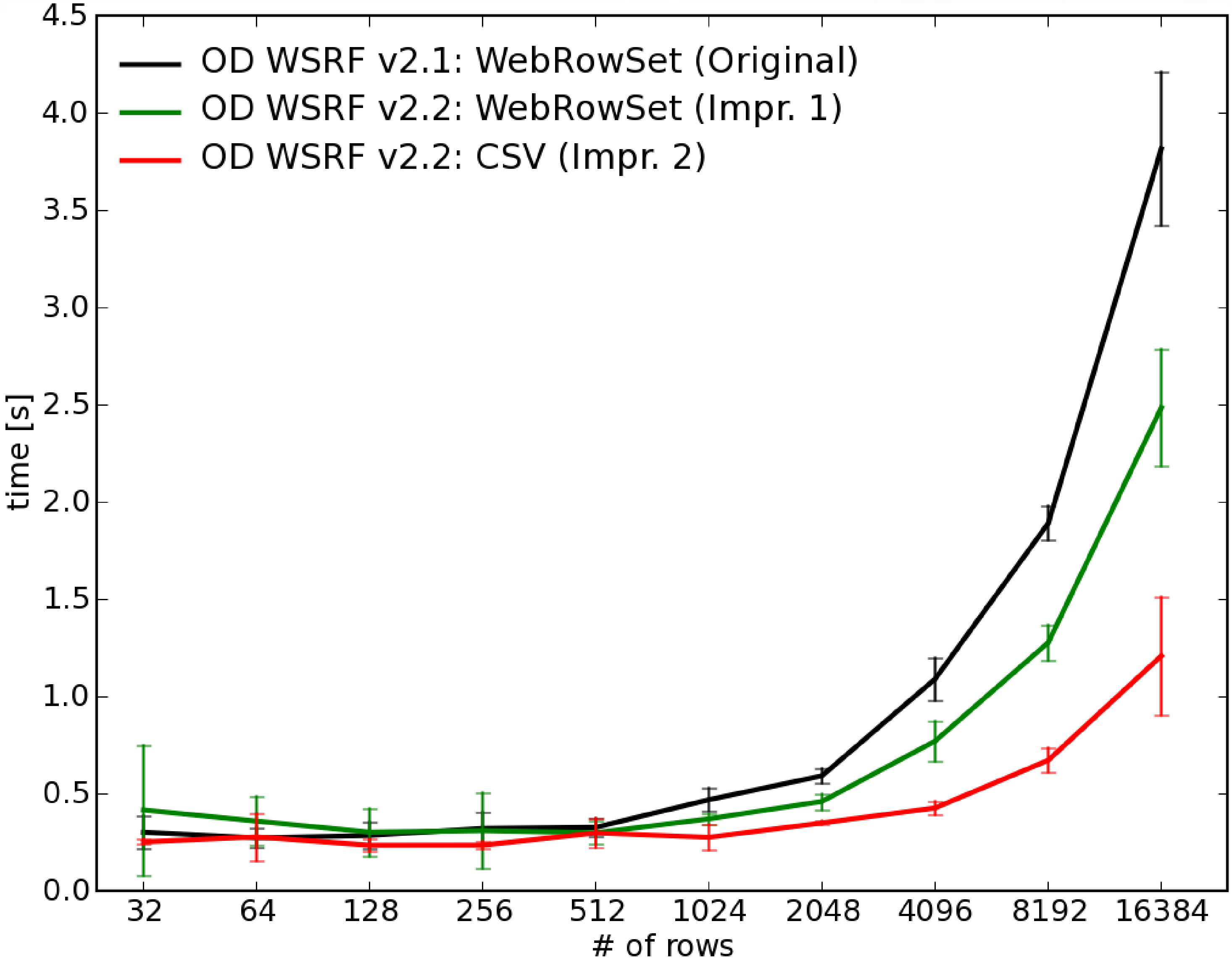
- LAN network packets traversed two routers.
  - Average network bandwidth 94 Mbits/s
  - Average round-trip latency <1 ms

Database

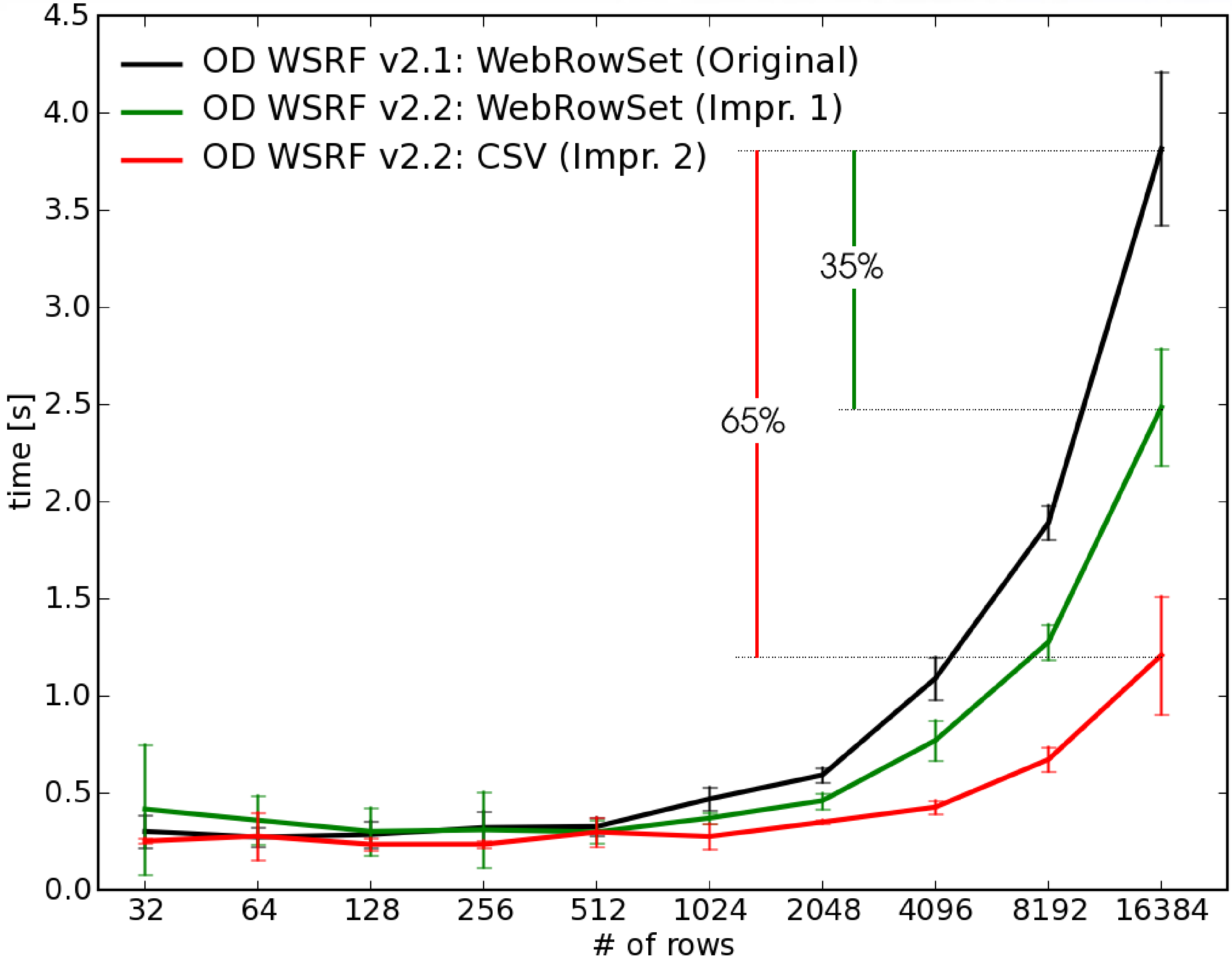
- MySQL 5.0.15
  - MySQL Connector/J ver. 3.1.10
  - Mean table row length (text) used in experiments was 66 bytes

- JVMs were warmed up before taking measurements.
- Results reported are the average of these runs
- Error bars indicating +/- standard deviation

Performance: Client + Server



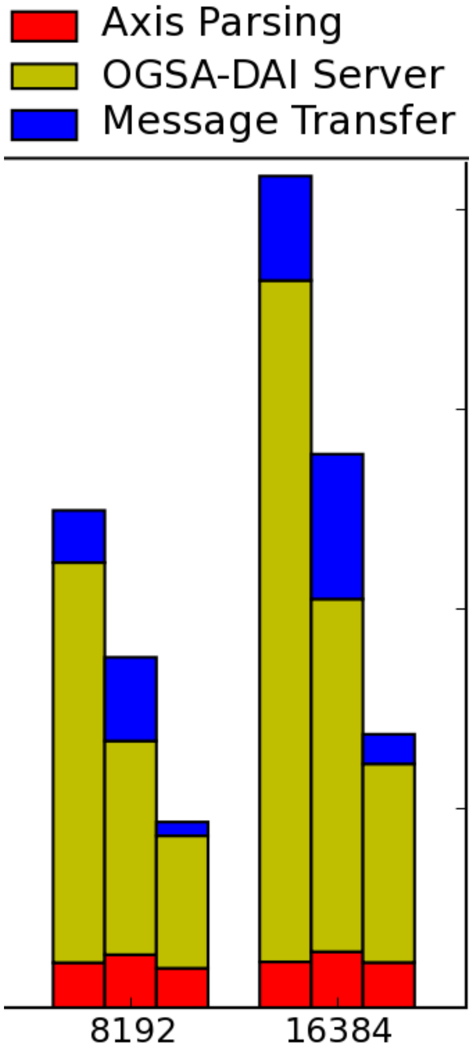
Performance: Client + Server



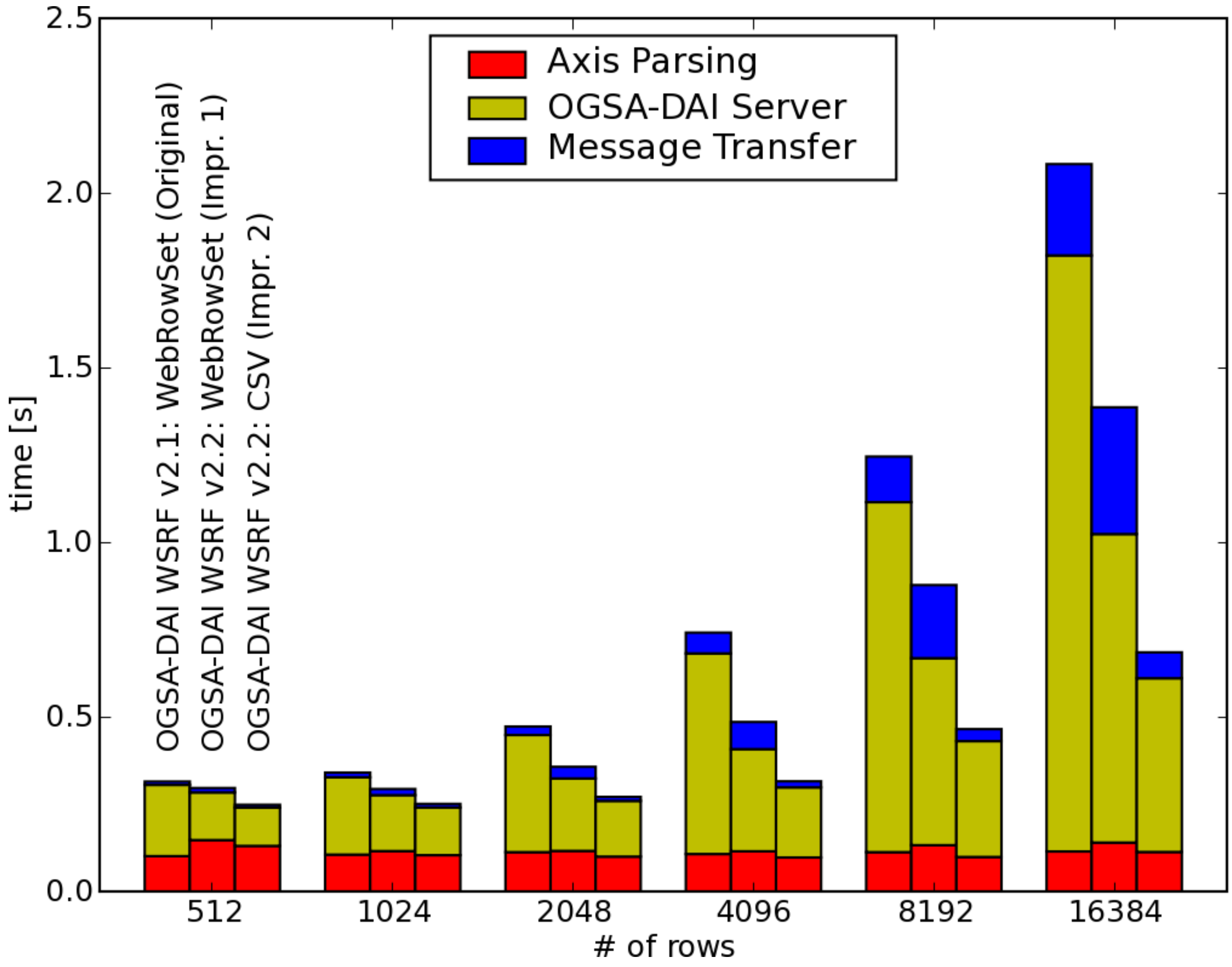
Server side time split

- Used Apache Axis `org.apache.axis.TIME` log category
- Records the time to execute incoming message
- Axis splits time into preamble, invoke, post and send phases
- In our plots

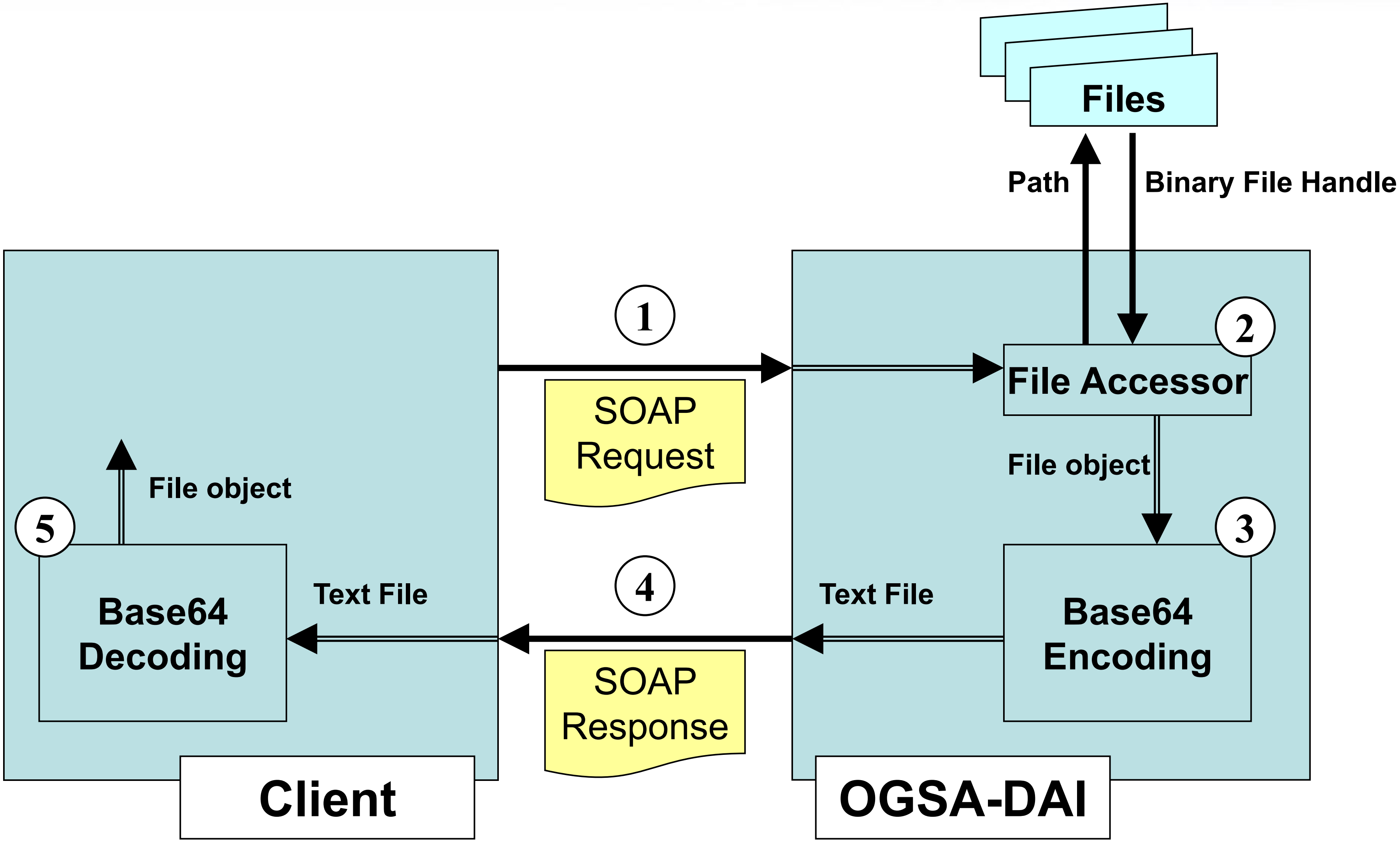
Axis Parsing = preamble
OGSA-DAI Server = invoke
Message Transfer = post + send



Performance: Server side details



Use Pattern 2: Transferring Binary Data



Improvement 3

Bottleneck

- Binary data needs to be Base64 encoded
  - Necessary to be included in a SOAP message
- Encoding and decoding requires additional computation
- The size of a data to be transferred grows by approximately 35%.
  - Base64 encoding uses 4 ASCII characters to represent 3 bytes

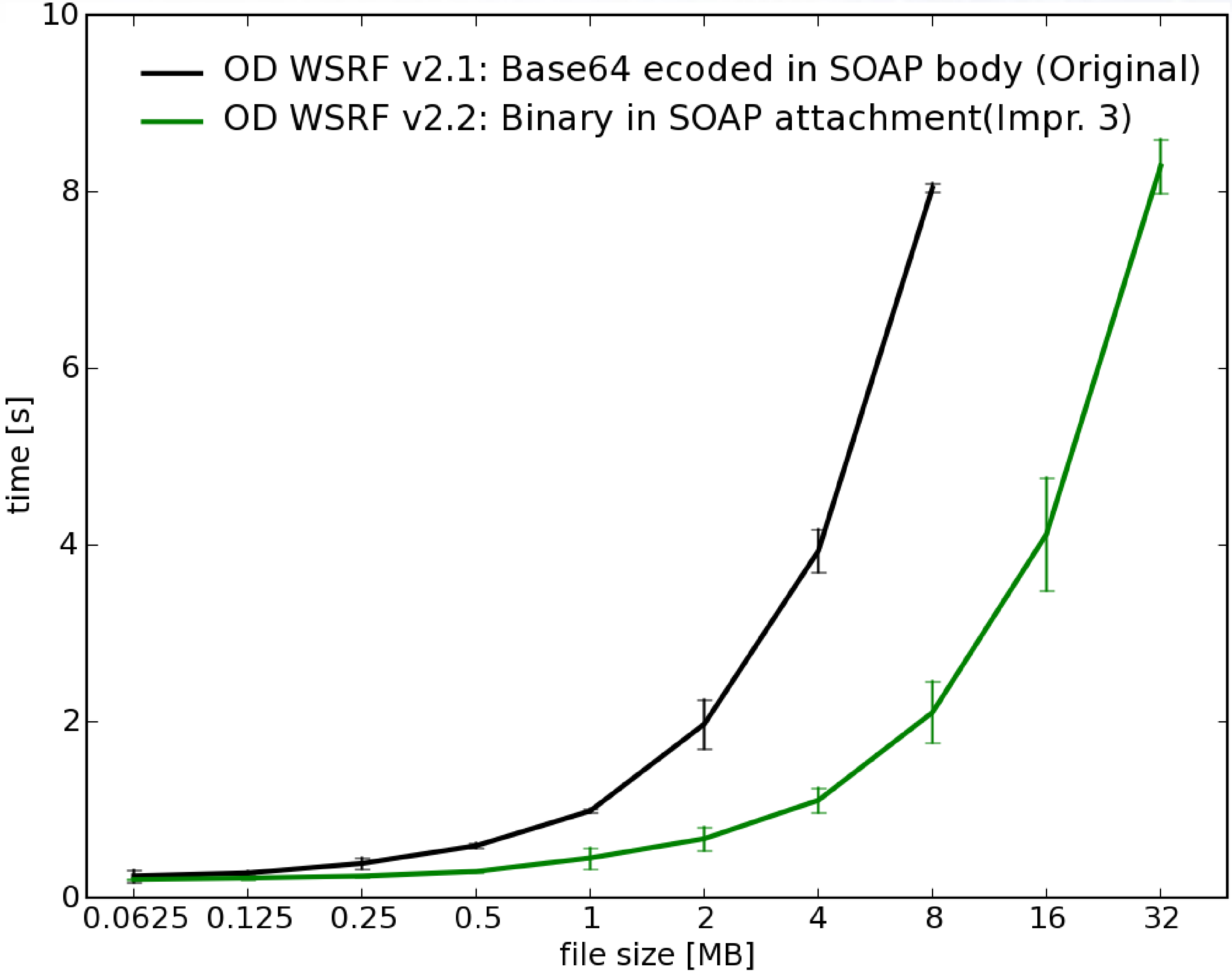
Improvement

- Both concerns addressed by using SOAP messages with attachments
  - No special encoding needed for binary data attached to a SOAP message

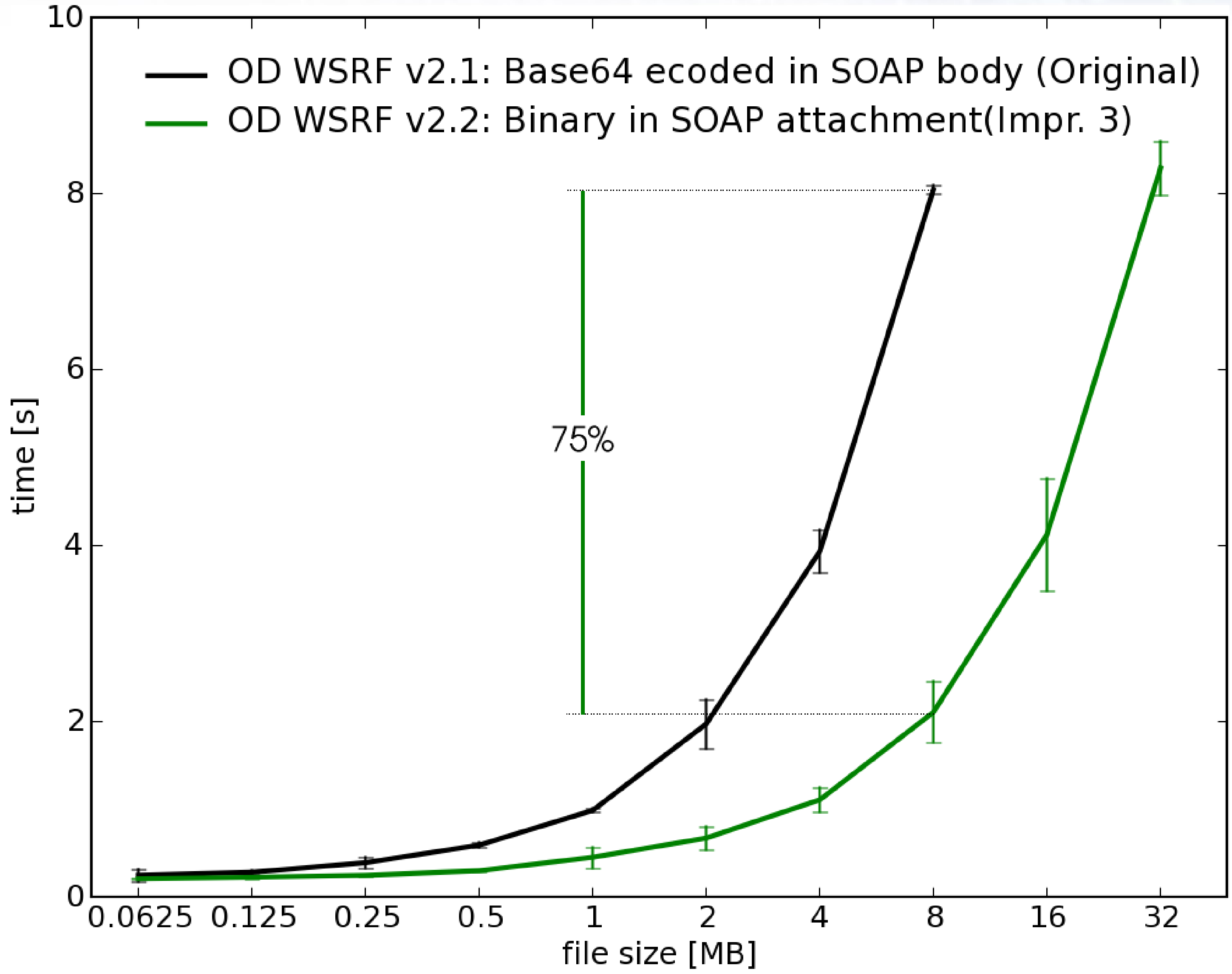
Drawback

- SOAP messages with attachments is not a standard feature of all SOAP engines
- This may affect interoperability

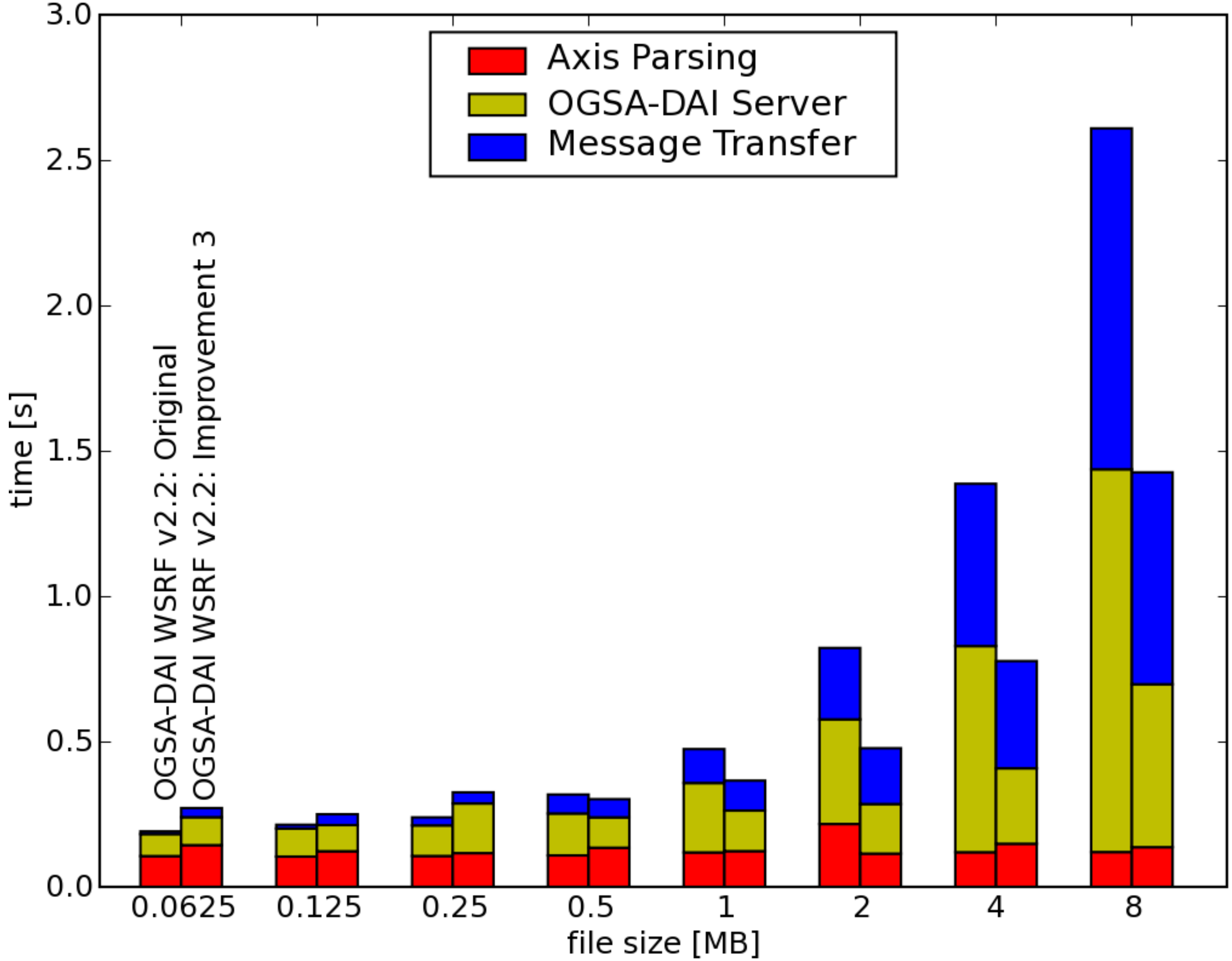
Performance: Client + Server



Performance: Client + Server



Performance: Server side details



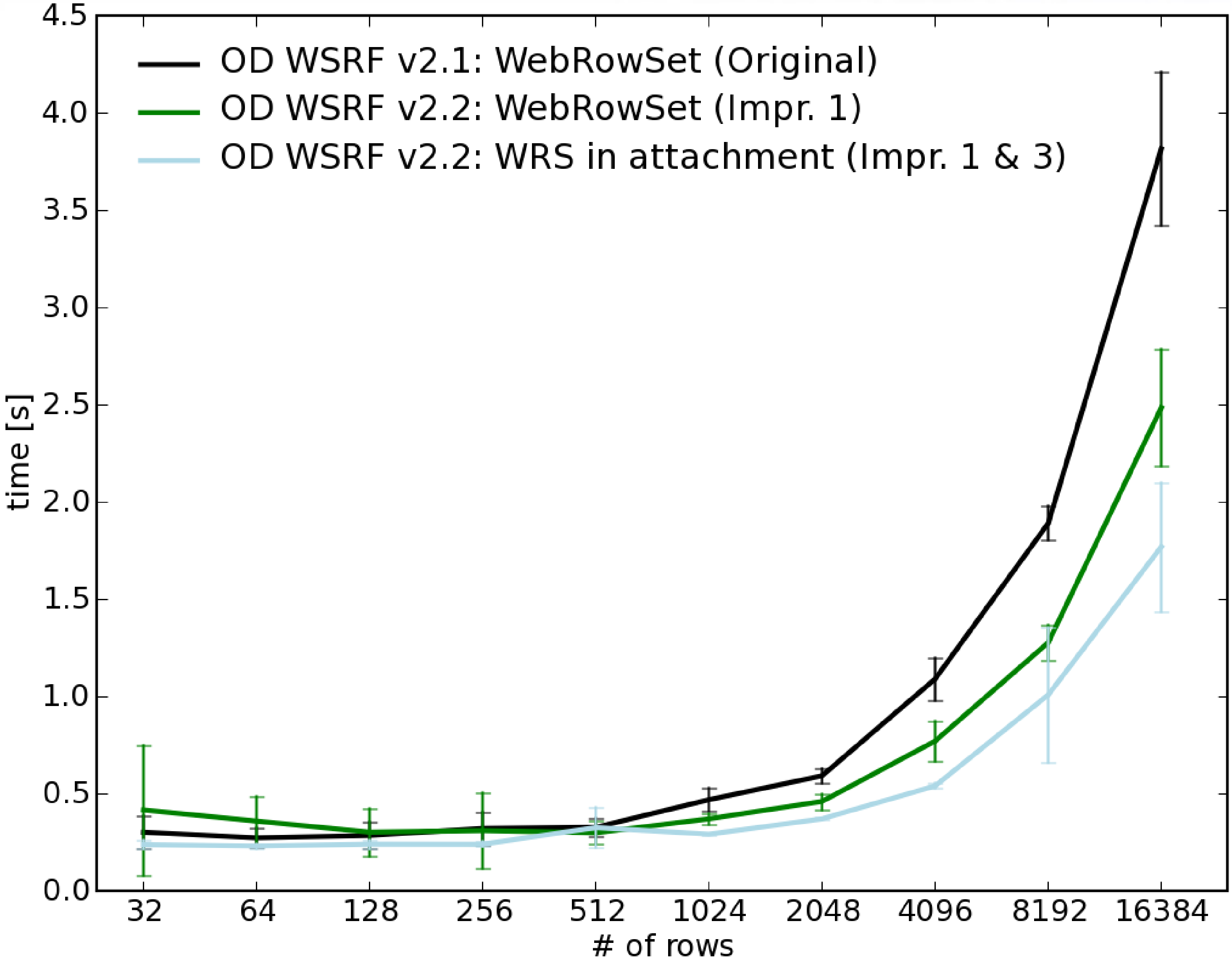
Delivering SQL Results as attachments

- Would expect to see additional improvement when delivering SQL Query results in attachments
  - SOAP message is smaller and easier to parse

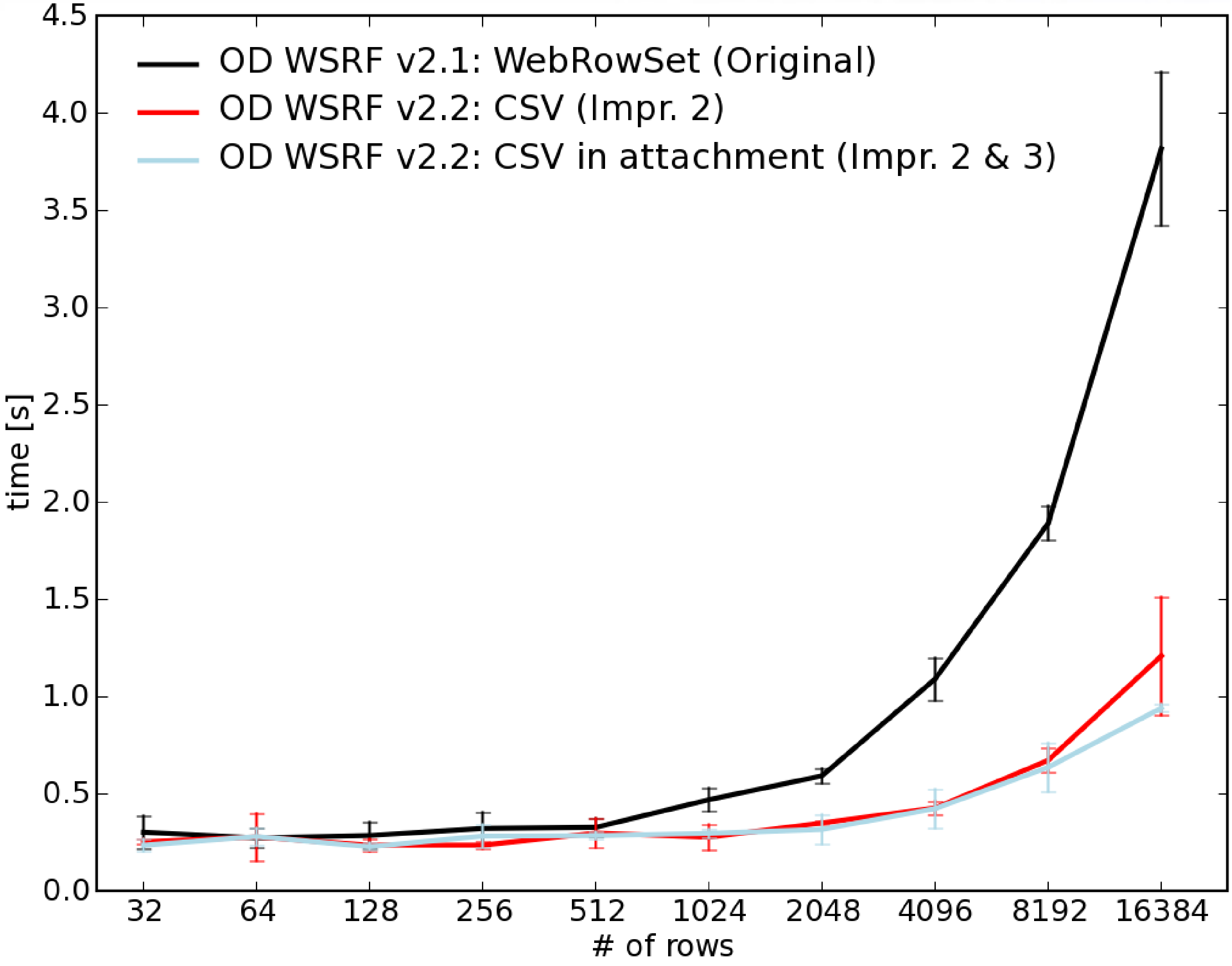
- Last experiments tested if we gain performance when we
  - Transfer WebRowSet documents as SOAP attachments
  - Transfer CSV documents as SOAP attachments

- In these experiments we test combined impact of all introduced improvements

Performance: Client + Server



Performance: Client + Server



Conclusions

- Status summary of an ongoing process to improve the OGSA-DAI performance
- Have analysed two typical use patterns:
  - These were profiled
  - Results used to implement a set of performance improvements
- Benefit demonstrated by comparing the performance of:
  - Current OGSA-DAI release (WSRF 2.2)
  - Previous OGSA-DAI release (WSRF 2.1)
- For the SQL use case reduced execution time by 65% by:
  - Optimising conversion routines
  - Using CSV format instead of WebRowSet
- SOAP with attachments gave a 75% improvement (for 8MB)
  - Significant reduction in the time needed to deliver binary data

General lessons learned

- Start by optimising conversion routines in your code
  - Especially if these are used often
- Profile your client and server code
  - Java profilers using Java Tool Interface (J2SE 5.0) are very powerful
  - Profiler manufactures often offer free licenses to open source projects
  - Results may surprise you!!
- Avoid using regular expressions for replacing characters
  - When called iteratively, accumulated cost may be significant
  - Writing dedicated parsers is usually easy and benefits are great
- Do not feel forced to use XML document formats
  - XML versatile but can be expensive in terms of space and processing
  - Use more lightweight formats when you do not need versatility
- Use SOAP with attachments to transfer binary data
  - And other large documents

Acknowledgements

People Involved:

- Mario Antonioletti
- Ally Hume
- Jen Schopf
- The OGSA-DAI Team
  - Author’s email: bartosz@epcc.ed.ac.uk
  - Paper available from: http://www.allhands.org.uk/2006/proceedings/

- This work is supported by the UK e-Science Grid Core Programme, through the Open Middleware Infrastructure Institute, and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38.
- We also gratefully acknowledge the input of our past and present partners and contributors to the OGSA-DAI project including: EPCC, IBM UK, IBM Corp., NeSC, University of Manchester, University of Newcastle and Oracle UK.