



#MongoDB

Aggregation Framework

Jason Zucchetto

Consulting Engineer, 10gen

Agenda

- State of Aggregation
- Pipeline
- Usage and Limitations
- Optimization
- Sharding
- Expressions (time permitting)
- Looking Ahead

State of Aggregation

State of Aggregation

- We're storing our data in MongoDB
- We need to do ad-hoc reporting, grouping, common aggregations, etc.
- What are we using for this?



Data Warehousing

Data Warehousing

- SQL for reporting and analytics
- Infrastructure complications
 - Additional maintenance
 - Data duplication
 - ETL processes
 - Real time?



<http://www.swissknifeshop.com/swiss-army-giant-by-wenger>

MapReduce

MapReduce

- Extremely versatile, powerful
- Intended for complex data analysis
- Overkill for simple aggregation tasks
 - Averages
 - Summation
 - Grouping

MapReduce in MongoDB

- Implemented with JavaScript
 - Single-threaded
 - Difficult to debug
- Concurrency
 - Appearance of parallelism
 - Write locks



<http://www.victorinox.com/us/product/Swiss-Army-Knives/Category/Classics/Classic-SD/53001>

Aggregation Framework

Aggregation Framework

- Declared in JSON, executes in C++
- Flexible, functional, and *simple*
 - Operation pipeline
 - Computational expressions
- Plays nice with sharding

Pipeline

Pipeline

- Process a stream of documents
 - Original input is a collection
 - Final output is a result document
- Series of operators
 - Filter or transform data
 - Input/output chain

```
ps ax | grep mongod | head -n 1
```

Pipeline Operators

- \$match
- \$project
- \$group
- \$unwind
- \$sort
- \$limit
- \$skip

Our Example Data

```
{  
  _id: 375,  
  title: "The Great Gatsby",  
  ISBN: "9781857150193",  
  available: true,  
  pages: 218,  
  chapters: 9,  
  subjects: [  
    "Long Island",  
    "New York",  
    "1920s"  
,  
  language: "English"  
}
```

\$match

- Filter documents
- Uses existing query syntax
- No geospatial operations or \$where

Matching Field Values

```
{  
  title: "The Great Gatsby",  
  pages: 218,  
  language: "English"  
}  
{
```

```
  title: "War and Peace",  
  pages: 1440,  
  language: "Russian"  
}
```

```
{  
  title: "Atlas Shrugged",  
  pages: 1088,  
  language: "English"  
}
```



```
{ $match: {  
  language: "Russian"  
}}
```



```
{  
  title: "War and Peace",  
  pages: 1440,  
  language: "Russian"  
}
```

Matching with Query Operators

```
{  
  title: "The Great Gatsby",  
  pages: 218,  
  language: "English"  
}
```

```
{  
  title: "War and Peace",  
  pages: 1440,  
  language: "Russian"  
}
```

```
{  
  title: "Atlas Shrugged",  
  pages: 1088,  
  language: "English"  
}
```



```
{ $match: {  
  pages: { $gt: 1000 }  
}}
```



```
{  
  title: "War and Peace",  
  pages: 1440,  
  language: "Russian"  
}
```

```
{  
  title: "Atlas Shrugged",  
  pages: 1088,  
  language: "English"  
}
```

\$project

- Reshape documents
- Include, exclude or rename fields
- Inject computed fields
- Create sub-document fields

Including and Excluding Fields

```
{  
  _id: 375,  
  title: "Great Gatsby",  
  ISBN: "9781857150193",  
  available: true,  
  pages: 218,  
  subjects: [  
    "Long Island",  
    "New York",  
    "1920s"  
,  
  language: "English"  
}
```



```
{ $project: {  
  _id: 0,  
  title: 1,  
  language: 1  
}}
```



```
{  
  title: "Great Gatsby",  
  language: "English"  
}
```

Renaming and Computing Fields

```
{  
  _id: 375,  
  title: "Great Gatsby",  
  ISBN: "9781857150193",  
  available: true,  
  pages: 218,  
  chapters: 9,  
  subjects: [  
    "Long Island",  
    "New York",  
    "1920s"  
  ],  
  language: "English"  
}
```



```
{ $project: {  
  avgChapterLength: {  
    $divide: ["$pages",  
              "$chapters"]  
  },  
  lang: "$language"  
}}
```



```
{  
  _id: 375,  
  avgChapterLength: 24.2222,  
  lang: "English"  
}
```

Creating Sub-Document Fields

```
{  
  _id: 375,  
  title: "Great Gatsby",  
  ISBN: "9781857150193",  
  available: true,  
  pages: 218,  
  subjects: [  
    "Long Island",  
    "New York",  
    "1920s"  
,  
  language: "English"  
}
```



```
{ $project: {  
  title: 1,  
  stats: {  
    pages: "$pages",  
    language: "$language",  
  }  
}}
```

```
{  
  _id: 375,  
  title: "Great Gatsby",  
  stats: {  
    pages: 218,  
    language: "English"  
  }  
}
```

\$group

- Group documents by an ID
 - Field reference, object, constant
- Other output fields are computed
 - \$max, \$min, \$avg, \$sum
 - \$addToSet, \$push
 - \$first, \$last
- Processes all data in memory

Calculating an Average

```
{  
  title: "The Great Gatsby",  
  pages: 218,  
  language: "English"  
}
```

```
{  
  title: "War and Peace",  
  pages: 1440,  
  language: "Russian"  
}
```

```
{  
  title: "Atlas Shrugged",  
  pages: 1088,  
  language: "English"  
}
```



```
{ $group: {  
  _id: "$language",  
  avgPages: { $avg:  
    "$pages" }  
}}
```



```
{  
  _id: "Russian",  
  avgPages: 1440  
}
```

```
{  
  _id: "English",  
  avgPages: 653  
}
```

Summating Fields and Counting

```
{  
  title: "The Great Gatsby",  
  pages: 218,  
  language: "English"  
}
```

```
{  
  title: "War and Peace",  
  pages: 1440,  
  language: "Russian"  
}
```

```
{  
  title: "Atlas Shrugged",  
  pages: 1088,  
  language: "English"  
}
```



```
{ $group: {  
  _id: "$language",  
  numTitles: { $sum: 1 },  
  sumPages: { $sum: "$pages" }  
}}
```



```
{  
  _id: "Russian",  
  numTitles: 1,  
  sumPages: 1440  
}
```

```
{  
  _id: "English",  
  numTitles: 2,  
  sumPages: 1306  
}
```

Collecting Distinct Values

```
{  
  title: "The Great Gatsby",  
  pages: 218,  
  language: "English"  
}
```

```
{  
  title: "War and Peace",  
  pages: 1440,  
  language: "Russian"  
}
```

```
{  
  title: "Atlas Shrugged",  
  pages: 1088,  
  language: "English"  
}
```



```
{ $group: {  
  _id: "$language",  
  titles: { $addToSet: "$title" }  
}}
```



```
{  
  _id: "Russian",  
  titles: [ "War and Peace" ]  
}
```

```
{  
  _id: "English",  
  titles: [  
    "Atlas Shrugged",  
    "The Great Gatsby"  
  ]  
}
```

\$unwind

- Operate on an array field
- Yield new documents for each array element
 - Array replaced by element value
 - Missing/empty fields → no output
 - Non-array fields → error
- Pipe to \$group to aggregate array values

Yielding Multiple Documents from One

```
{  
  title: "The Great Gatsby",  
  ISBN: "9781857150193",  
  subjects: [  
    "Long Island",  
    "New York",  
    "1920s"  
  ]  
}
```



```
{ $unwind: "$subjects" }
```

```
{  
  title: "The Great Gatsby",  
  ISBN: "9781857150193",  
  subjects: "Long Island"  
}  
{  
  title: "The Great Gatsby",  
  ISBN: "9781857150193",  
  subjects: "New York"  
}  
{  
  title: "The Great Gatsby",  
  ISBN: "9781857150193",  
  subjects: "1920s"  
}
```



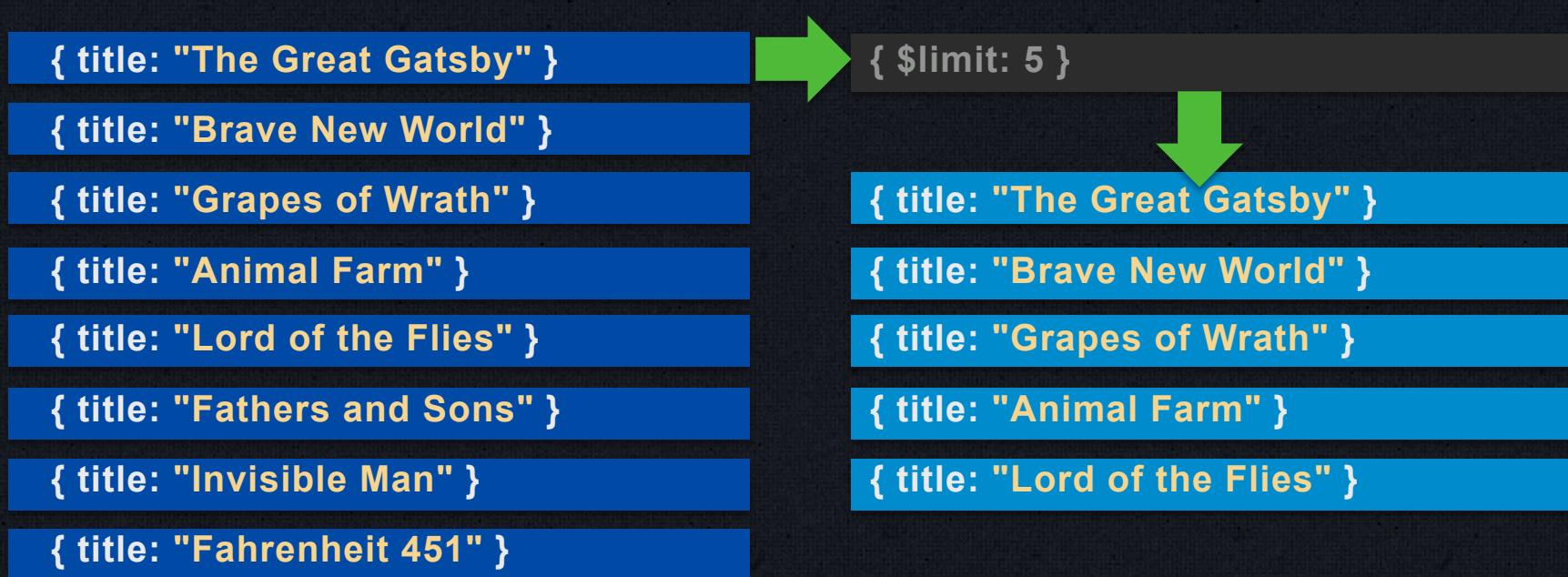
\$sort, \$limit, \$skip

- Sort documents by one or more fields
 - Same order syntax as cursors
 - Waits for earlier pipeline operator to return
 - In-memory unless early and indexed
- Limit and skip follow cursor behavior

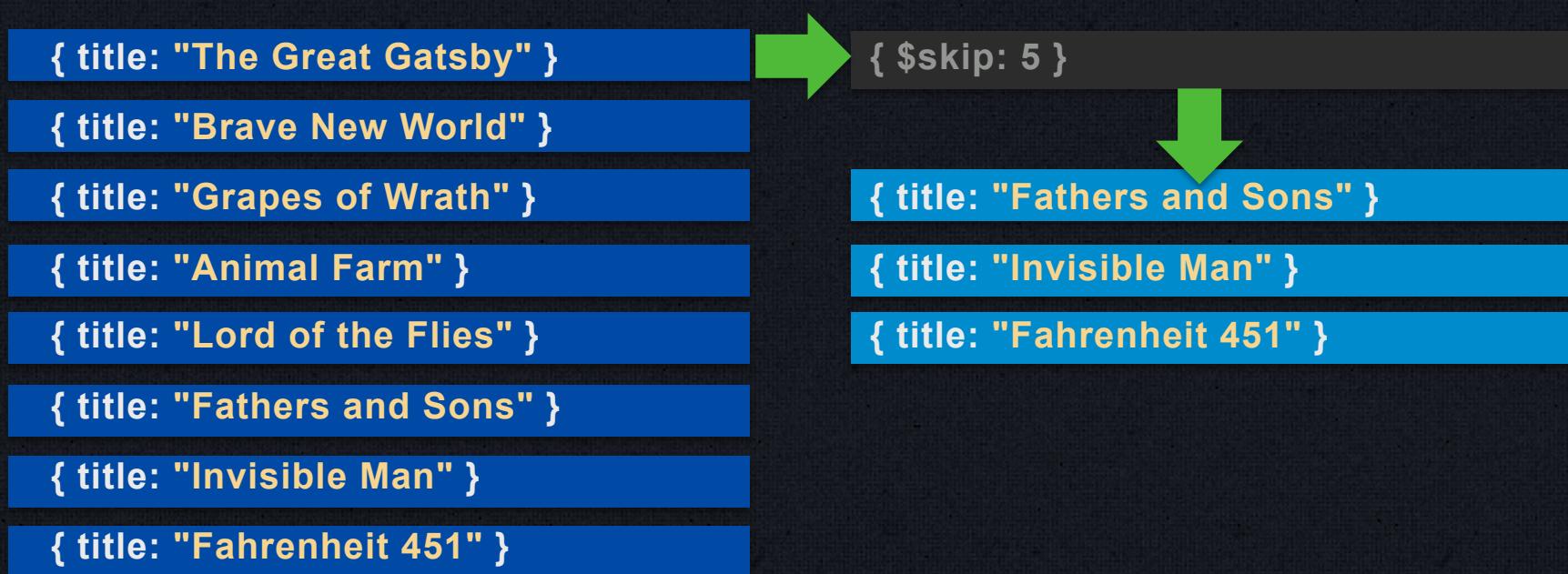
Sort All the Documents in the Pipeline



Limit Documents Through the Pipeline



Skip Over Documents in the Pipeline



Usage and Limitations

Usage

- `collection.aggregate()` method
 - Mongo shell
 - Most drivers
- `aggregate` database command

Collection

```
db.books.aggregate([
  { $project: { language: 1 }},
  { $group: { _id: "$language", numTitles: { $sum: 1 }}}
])
{
  result: [
    { _id: "Russian", numTitles: 1 },
    { _id: "English", numTitles: 2 }
  ],
  ok: 1
}
```



Database Command

```
db.runCommand({  
    aggregate: "books",  
    pipeline: [  
        { $project: { language: 1 }},  
        { $group: { _id: "$language", numTitles: { $sum: 1 }}}  
    ]  
})  
  
{  
    result: [  
        { _id: "Russian", numTitles: 1 },  
        { _id: "English", numTitles: 2 }  
    ],  
    ok: 1  
}
```



Limitations

- Result limited by BSON document size
 - Final command result
 - Intermediate shard results
- Pipeline operator memory limits
- Some BSON types unsupported
 - Binary, Code, deprecated types

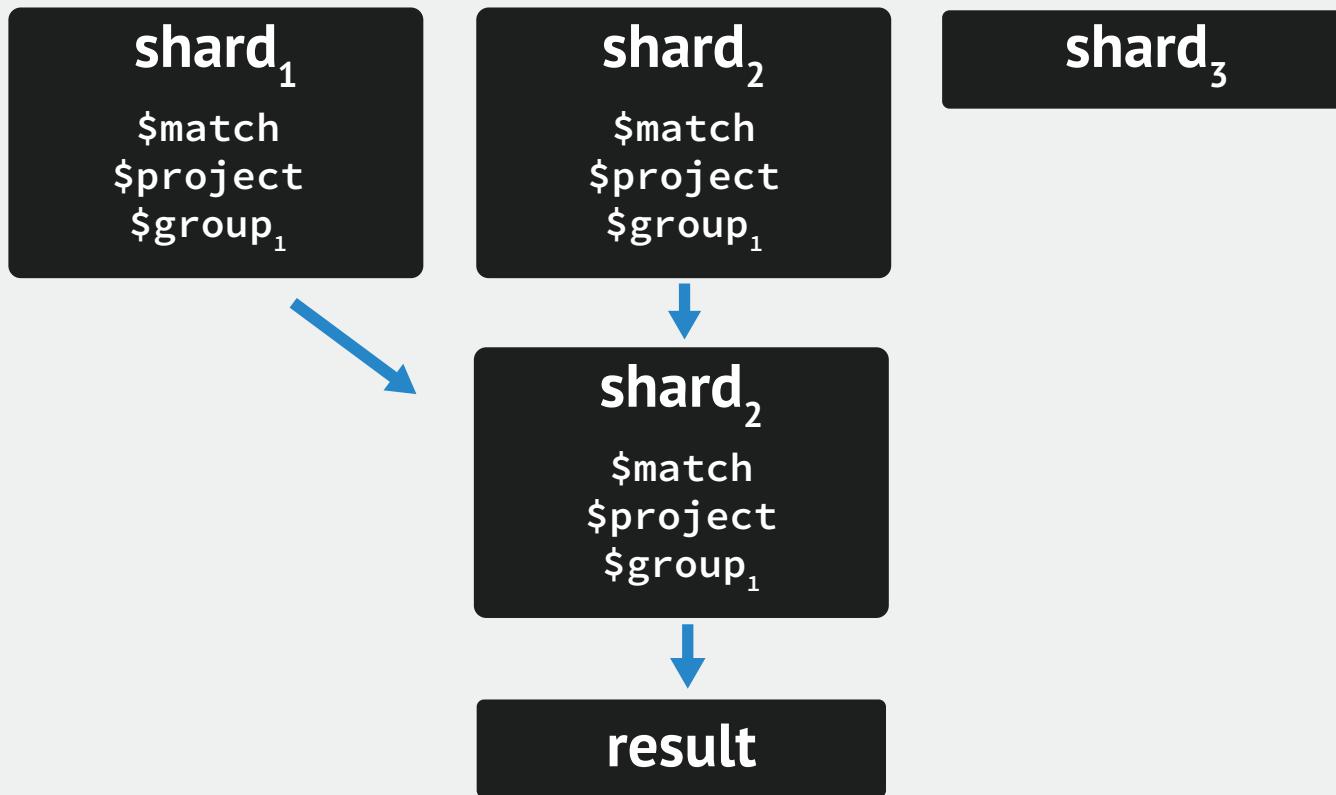
Sharding

Sharding

- Split the pipeline at first \$group or \$sort
 - Shards execute pipeline up to that point
 - mongos merges results and continues
- Early \$match may excuse shards
- CPU and memory implications for mongos

Sharding

```
[  
  { $match: { /* filter by shard key */ }},  
  { $project: { /* select fields */ }},  
  { $group: { /* group by some field */ }},  
  { $sort: { /* sort by some field */ }},  
  { $project: { /* reshape result */ }}]  
]
```



Sharding

Framework Use Cases

- Basic aggregation queries
- Ad-hoc reporting
- Real-time analytics
- Visualizing time series data

Extending the Framework

- Adding new pipeline operators, expressions
- \$out and \$tee for output control
 - <https://jira.mongodb.org/browse/SERVER-3253>

Future Enhancements

- Automatically move \$match earlier if possible
- Pipeline explain facility
- Memory usage improvements
 - Grouping input sorted by _id
 - Sorting with limited output

Enabling Developers

- Doing more within MongoDB, faster
- Refactoring MapReduce and groupings
 - Replace pages of JavaScript
 - Longer aggregation pipelines
- Quick aggregations from the shell



#MongoDB

Thank You

Jason Zucchetto

Consulting Engineer, 10gen