Aggregation Framework

Derick Rethans—derick@10gen.com—@derickr

10gen | the MongoDB company

mongoDB

Agenda

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

Stage 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
  - Extract Transform Load (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" : "w94069675",
        "tags" : {
                "addr:housenumber" : "64",
                "addr:street" : "Borough High Street",
                "amenity" : "restaurant",
                "building" : "yes",
                "cuisine" : "indian",
                "name" : "Truly Indian"
        },
}
```

$match

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

Matching Field Values

```
{
  "_id" : "w42931536",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "french",
    "name" : "Le Comptoir Gascon",
  }
}
{
  "_id" : "w50732182",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "burger",
    "name" : "Starving Marvin's",
  }
}
{
  "_id" : "w58537927",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "chinese",
    "name" : "Royal Garden Restaurant"
  }
}
```

$\rightarrow$

```
{
  $match: {
    "tags.cuisine": "chinese"
  }
}
```

```
{
  "_id" : "w58537927",
  "tags" : {
    "building" : "yes",
    "cuisine" : "chinese",
    "name" : "Royal Garden Restaurant"
  }
}
```

Matching with Query Operators

```
{
  "_id" : "n1240135999",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "thai",
    "name" : "Addie's Thai"
  }
}
{
  "_id" : "n1242469688",
  "tags" : {
    "addr:postcode" : "CR6 9EG",
    "addr:street" : "Farleigh Road",
    "amenity" : "restaurant",
    "cuisine" : "regional",
    "name" : "The Horseshoe"
  }
}
{
  "_id" : "n1242549289",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "ethiopian",
    "name" : "Queen of Sheba"
  }
}
```

→

```
{  $match: {
  "tags.addr:postcode": { $exists: true }
} }
```

```
{
  "_id" : "n1242469688",
  "tags" : {
    "addr:postcode" : "CR6 9EG",
    "addr:street" : "Farleigh Road",
    "amenity" : "restaurant",
    "cuisine" : "regional",
    "name" : "The Horseshoe"
  }
}
```

$project

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

Including and Excluding Fields

```
{
  "_id" : "n47733711",
  "type" : NumberLong(1),
  "loc" : [
    0.2344868,
    51.5938555
  ],
  "tags" : {
    "amenity" : "fast_food",
    "cuisine" : "fish_and_chips",
    "name" : "The Frying Pan"
  }
},
```

$\longrightarrow$

```
{
    $project: { _id: 0, tags: 1 }
}
```

```
{
  "tags" : {
    "amenity" : "fast_food",
    "cuisine" : "fish_and_chips",
    "name" : "The Frying Pan"
  }
},
```

Renaming and Computing Fields

```
{
  "_id" : "n375475617",
  "type" : NumberLong(1),
  "loc" : [
    -0.1502324,
    51.4440935
  ],
  "tags" : {
    "addr:housenumber" : "37",
    "addr:street" : "Bedford Hill",
    "amenity" : "restaurant",
    "cuisine" : "greek",
    "name" : "Meze Kitchen"
  }
},
```

$\longrightarrow$

```
{ $project: {
  address: { $concat: [
    '$tags.addr:housenumber',
    ' ',
    '$tags.addr:street'
  ] },
  'desc': '$tags'
} }
```

```
{
  "_id" : "n375475617",
  "address" : "37 Bedford Hill",
  "desc" : {
    "addr:housenumber" : "37",
    "addr:street" : "Bedford Hill",
    "amenity" : "restaurant",
    "cuisine" : "greek",
    "name" : "Meze Kitchen"
  }
},
```

Reformat Sub-Documents

```
{
  "_id" : "n691721239",
  "type" : NumberLong(1),
  "loc" : [
    -0.135621,
    51.515325
  ],
  "tags" : {
    "addr:city" : "London",
    "addr:country" : "GB",
    "addr:housenumber" : "187b",
    "addr:street" : "Wardour Street",
    "amenity" : "restaurant",
    "cuisine" : "korean",
    "name" : "Red Devil"
  }
}
```

⟶

```
{ $project: {
  name:    '$tags.name',
  cuisine: '$tags.cuisine',
  address: {
    nr:     '$tags.addr:housenumber',
    street: '$tags.addr:street',
    city:   '$tags.addr:city'
  }
} }
```

```
    {
      "_id" : "n691721239",
      "name" : "Red Devil",
      "cuisine" : "korean",
      "address" : {
        "nr" : "187b",
        "street" : "Wardour Street",
        "city" : "London"
      }
    }
```

$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

Summating Fields and Counting

```
{
  "_id" : "n306036143",
  "tags" : {
    "amenity" : "cafe",
    "cuisine" : "italian",
    "name" : "Saponara"
  }
},
{
  "_id" : "n306042529",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "asian",
    "layer" : "2",
    "name" : "Wagamama"
  }
},
{
  "_id" : "n306042546",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "sushi",
    "layer" : "2",
    "name" : "Yo! Sushi"
  }
}
```

$\longrightarrow$

```
{ $group: {
  _id: '$tags.amenity',
  count: { '$sum': 1 }
} }
```

```
{
  "_id" : "restaurant",
  "count" : 2
},
{
  "_id" : "cafe",
  "count" : 1
},
```

Collecting Distinct Values

```
{
  "_id" : "n388320891",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "chinese",
    "name" : "China GaGa"
  }
},
{
  "_id" : "n388826644",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "japanese",
    "name" : "Oisi"
  }
},
{
  "_id" : "n389031406",
  "tags" : {
    "amenity" : "pub",
    "cuisine" : "tapas",
    "food" : "yes",
    "name" : "Tonic"
  }
}
```

→

```
{ $group: {
  _id: '$tags.amenity',
  cuisines: { '$addToSet': '$tags.cuisine' }
} }
```

```
{
  "_id" : "pub",
  "cuisines" : [
    "tapas"
  ]
},
{
  "_id" : "restaurant",
  "cuisines" : [
    "japanese",
    "chinese"
  ]
}
```

Grouping with a Compound Key

```
{
  "_id" : "n322270228",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "italian",
    "name" : "Ristorante LA COLLINA",
  }
},
{
  "_id" : "n322270302",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "indian",
    "name" : "The Connoisseur",
  }
},
{
  "_id" : "n322270304",
  "tags" : {
    "amenity" : "restaurant",
    "cuisine" : "italian",
    "deli" : "yes",
    "name" : "Incanto",
  }
}
```

$\longrightarrow$

```
{ $group: {
  _id: {
    'amenity' : '$tags.amenity',
    'cuisine' : '$tags.cuisine'
  }
  names: { '$addToSet': '$tags.name' }
} }
```

```
{
  "_id" : {
    "amenity" : "restaurant",
    "cuisine" : "indian"
  },
  "names" : [
    "The Connoisseur"
  ]
},
{
  "_id" : {
    "amenity" : "restaurant",
    "cuisine" : "italian"
  },
  "names" : [
    "Incanto",
    "Ristorante LA COLLINA"
  ]
}
```

$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

```
{
  "_id" : "n476033502",
  "type" : NumberLong(1),
  "tags" : [
    "addr:housenumber=125",
    "addr:street=Alban Gate",
    "amenity=restaurant",
    "cuisine=pizza",
    "level=1",
    "name=Pizza Express",
    "phone=020 7600 8880",
    "postcode=EC2Y 5AS"
  ]
}
```

→

```
{
  $unwind: '$tags'
}
```

```
{
  "_id" : "n476033502",
  "type" : NumberLong(1),
  "tags" : "addr:housenumber=125"
},
{
  "_id" : "n476033502",
  "type" : NumberLong(1),
  "tags" : "addr:street=Alban Gate"
},
{
  "_id" : "n476033502",
  "type" : NumberLong(1),
  "tags" : "amenity=restaurant"
},
{
  "_id" : "n476033502",
  "type" : NumberLong(1),
  "tags" : "cuisine=pizza"
},
…
```

$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 behaviour

Sort All the Documents in the Pipeline

```
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Yard" },
{ "name" : "Le Sacré Coeur" },
{ "name" : "Ask" },
{ "name" : "Vigata" },
{ "name" : "Sedir" },
{ "name" : "Parveen" },
{ "name" : "Mem & Laz" },
{ "name" : "Gallipoli" }
```

$\longrightarrow$

```
{ $sort: {
  name: 1
} }
```

```
{ "name" : "Ask" },
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Gallipoli" }
{ "name" : "Le Sacré Coeur" },
{ "name" : "Mem & Laz" },
{ "name" : "Parveen" },
{ "name" : "Sedir" },
{ "name" : "Vigata" },
{ "name" : "Yard" },
```

Limit Documents through the Pipeline

```
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Yard" },
{ "name" : "Le Sacré Coeur" },
{ "name" : "Ask" },
{ "name" : "Vigata" },
{ "name" : "Sedir" },
{ "name" : "Parveen" },
{ "name" : "Mem & Laz" },
{ "name" : "Gallipoli" }
```

→

```
{ $limit : 5 }
```

```
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Yard" },
{ "name" : "Le Sacré Coeur" },
{ "name" : "Ask" },
{ "name" : "Vigata" },
```

Skip Over Documents in the Pipeline

```
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Yard" },
{ "name" : "Le Sacré Coeur" },
{ "name" : "Ask" },
{ "name" : "Vigata" },
{ "name" : "Sedir" },
{ "name" : "Parveen" },
{ "name" : "Mem & Laz" },
{ "name" : "Gallipoli" }
```

$\longrightarrow$

```
{ $skip : 3 }
```

```
{ "name" : "Ask" },
{ "name" : "Vigata" },
{ "name" : "Sedir" },
{ "name" : "Parveen" },
{ "name" : "Mem & Laz" },
{ "name" : "Gallipoli" }
```

Order of $sort, $limit, $skip is Important

```
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Yard" },
{ "name" : "Le Sacré Coeur" },
{ "name" : "Ask" },
{ "name" : "Vigata" },
{ "name" : "Sedir" },
{ "name" : "Parveen" },
{ "name" : "Mem & Laz" },
{ "name" : "Gallipoli" }
```

→

```
[
  { $limit : 4 },
  { $sort : { name: 1 } }
]
```

```
{ "name" : "Ask" },
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Le Sacré Coeur" },
{ "name" : "Yard" },
```

```
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Yard" },
{ "name" : "Le Sacré Coeur" },
{ "name" : "Ask" },
{ "name" : "Vigata" },
{ "name" : "Sedir" },
{ "name" : "Parveen" },
{ "name" : "Mem & Laz" },
{ "name" : "Gallipoli" }
```

→

```
[
  { $sort : { name: 1 } },
  { $limit : 4 }
]
```

```
{ "name" : "Ask" },
{ "name" : "Blues Restaurant & Bar" },
{ "name" : "Gallipoli" }
{ "name" : "Le Sacré Coeur" },
```

Usage and Limitations

Usage

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

Collection Method

```
db.poi.aggregate( [
  { $match: { 'tags.amenity': 'restaurant' } },
  { $skip: 100 },
  { $limit: 1 }
] }
```

Result:

```
{
  "result" : [
    {
      "_id" : "n272078075",
      "type" : NumberLong(1),
      "tags" : {
        "amenity" : "restaurant",
        "cuisine" : "turkish",
        "name" : "Cyprus Mangal"
      }
    },
  ],
  "ok" : 1
}
```

Database Command

```
db.runCommand({
  aggregate: "poi",
  pipeline: [
  [
    { $match: { 'tags.amenity': 'restaurant' } },
    { $skip: 100 },
    { $limit: 1 }
  ]
});
```

Result:

```
{
  "result" : [
    {
      "_id" : "n272078075",
      "type" : NumberLong(1),
      "tags" : {
        "amenity" : "restaurant",
        "cuisine" : "turkish",
        "name" : "Cyprus Mangal"
      }
    },
  ],
  "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

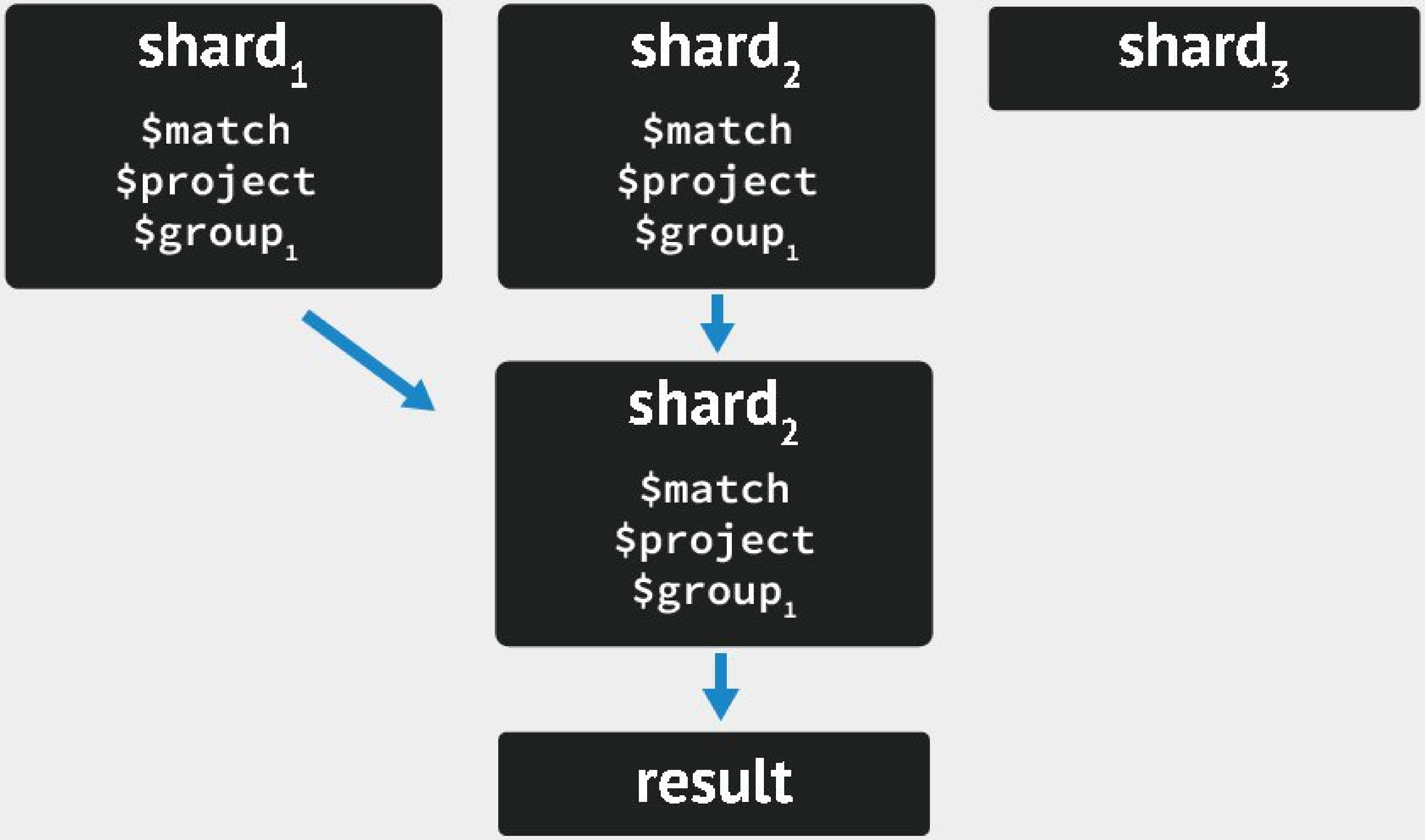
Sharding

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 */      } }
]
```



shard$_1$
$match
$project
$group$_1$
shard$_2$
$match
$project
$group$_1$
shard$_3$
shard$_2$
$match
$project
$group$_1$
result

Sharding

Looking Ahead

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
- Cursor support

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

Slides:
http://derickrethans.nl/talks/mongo-aggregation-stockholm13

Derick Rethans—@derickr—derick@10gen.com