From Templates to Beam: Chartboost’s Journey
About us

Austin Bennett

Ferran Fernandez
Chartboost

- ADS!

- “Build your mobile business with the leading in-app monetization and programmatic advertising platform”
Agenda

- Initial Beam Usage
- Config Templates: Pros and Cons
- Deeper Evaluation of Config Templates
- Findings, Ongoing-Changes, Future State
Initial Beam Usage
Chartboost's initial use of Beam

- **Moving from Apache Spark to Beam**
  - Decision to move away from Apache Spark due to legacy code and the desire to utilize Dataflow.
  - Transitioning streaming jobs first: Focus on processing JSON events from Kafka and writing into BigQuery.
  - Early realization: Code adaptability to batch processing was possible.

- **Language Choice and Team Dynamics**
  - Consideration of language options: Java selected for its features and performance.
  - Acknowledging the team's varying levels of comfort with Java.
  - Need for a collaborative solution to address team concerns and ensure successful implementation.
Config Templates

Combination of PTransform and PCollections

Or

PCollection → PTransform

Config Template
Config Templates

JSON Event

```json
{
    "user": "Ferran",
    "info": {
        "id": 1234
    },
    "device": "Android",
    "response": [
        {
            "web_name": "www.chartboost.com",
            "web_code": 200,
            "web_body": "chartboost"
        },
        {
            "web_name": "www.empty.com",
            "web_code": 403
        }
    ]
}
```

Config Template

```json
{
    "json_parse": {
        "user": {
            "bq_column": "user_name",
            "function": "fieldToLower(user)"
        },
        "info.id": {
            "bq_column": "user_id"
        },
        "device": {
            "bq_column": "device",
            "function": "fieldToLower(device)"
        },
        "filter": {
            "allowlist": "android"
        }
    },
    "web_name": {
        "bq_column": "web_name",
        "unnest_field": "response"
    },
    "web_code": {
        "bq_column": "web_code",
        "unnest_field": "response"
    },
    "web_body": {
        "bq_column": "web_body",
        "unnest_field": "response"
    }
}
```

functions:

```
function fieldToLower(a) {return a.toLowerCase();}
```
How Config Templates helped in the early stages

- **Easy to use common-pattern**
  - Simplifying data transformation for non-Java developers.
  - Seamless integration process.

- **Advantages & Considerations**
  - Enhanced pipeline reusability and maintainability over prior Spark solution.
  - Performance considerations.
Pros and Cons of Config Templates
Perceived Pros of Config Templates

- Easy and quick setup.
- Simplified pipeline development.
- Flexibility in customizing templates.
Perceived Cons of Config Templates

- Limited functionality.
- Potential challenges in scaling and performance optimization.
- Lack of fine-grained control over the pipeline.
Config Templates under evaluation
Experiments and Metrics

We evaluated the efficiency of our current setup vs pure Java:

- Mapping Elements:
  - Map the JSON attributes into BigQuery columns without any transformations

- Simple JavaScript:
  - Map the JSON attributes into BigQuery with simple transformations such as upper and lower case transformation.

- More complex JavaScript transformations:
  - Inclusion of recursive transformations on arrays and other complex data types.

https://cloud.google.com/community/tutorials/metrics-export-with-mql
Experiments and Metrics

https://cloud.google.com/community/tutorials/metrics-export-with-mql
### Experiment Results

<table>
<thead>
<tr>
<th>Experiment</th>
<th>CB Template CPU [Baseline]</th>
<th>Java CPU [Target]</th>
<th>CPU Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Elements</td>
<td>22.16%</td>
<td>18.02%</td>
<td>(18.68%)</td>
</tr>
<tr>
<td>Map + Transform</td>
<td>28.30%</td>
<td>22.98%</td>
<td>(18.8%)</td>
</tr>
<tr>
<td>Map + Complex Transform</td>
<td>29.44%</td>
<td>23.62%</td>
<td>(19.77%)</td>
</tr>
</tbody>
</table>

**Note:** 50 machines. N1-standard-8. No autoscale. 400vCPUs. Processing 1M elements/s.
Benefits of Writing Actual Beam Code

- Greater flexibility and control over pipeline logic.
- Improved performance and scalability.
- Enhanced customization options.
Changes under way
Towards Beam

- Current Config templates limited us to per record operations.
  - There are business use cases we can’t address with current config templates.

- Preference to less code that we maintain internally
  - We would be happy to explore “A Declarative API for Beam”

- Many of engineers were not familiar with Beam.
  - Though we’ve now gotten the experience.
Simpler [ esp. to maintain ] pipelines

(A) protobuff
Protocol Buffers

(B) Less private/internal code means less code we must maintain ourselves [ shared/community support! ]
Conclusion and Takeaways

- Config templates have a lot of positives.
  - ESP. LOTS of business value, and not high learning curve
  - Would be great if something like this existed “in” the OSS Project

- Beam learning curve was intimidating.
  - In practice, it was straightforward.

- We have decent scale, growing the team’s knowledge/abilities has been worthwhile
  - Ensure prepared to address more advanced use cases, and save compute costs
If you’re into benchmarking

Benchmarking Beam pipelines on Dataflow
In room: Palisades
At 17:15 TODAY

Or online, see in YT ( ?below?)

Pranav Bhandari
Community Discussion: Future of Beam

Do share some input ... What would you like to see in Beam's future?


Join us
At: 16:45
Room: Horizon
Chartboost is Hiring

https://www.chartboost.com/careers

We have 6 Engineers here at the Summit, and are happy to chat

[ not only about jobs, but we like technology :-) ]
QUESTIONS?