Chester “Chet” Weger

chester.weger@gmail.com || (909)706-2026 || chet-weger.herokuapp.com || linkdin.com/in/chet-weger || github.com/chetweger/

Career Summary

Extensive experience developing *scalable* and *performant* full-stack reporting and insights products that analyze over 50 billion transaction logs generated by Rubicon Project’s *AdTech marketplace* per day. Performed maintenance on a legacy OLAP reporting product designed and architected using *MySQL* and a *snowflake schema* with hourly batch ingestion from Hadoop jobs. Helped develop the next generation *performance reporting* product architected using the Druid datastore and a *star schema*. Contributed to a suite of insights applications capable of providing ***insights*** such as “*How much **revenue** will be **gained** by unblocking advertiser “Ebay” on site amazon.com?”* Rewrote several legacy Hadoop jobs using Spark and Scala, and developed new data-pipelines using Spark.

Highlights

- Won Internal Hackathon “Automation Prize” – used Sankey chart to visualize the flow of money among marketplace entities.
- Vertx.x *Rabbit Holing* – traced a Heisenbug in production to vertx.x hardcoding the maximum size of an HTTP Header causing bug when users-agent was getting cookie overflow from the *rubiconproject.com* domain.
- MongoDB Fiasco – patched an urgent fire preventing output from Hadoop job from successfully being inserted into MongoDB caused by documents exceeding the maximum BSON document size. Used GZIP to compress problematically large bid-distribution objects to roughly *10%* of the original size.
- Using the Power of Druid – used Druid’s Cascading Extraction Function to add filterable columns representing many-to-many relationships, helping a legacy reporting product to be decommissioned.
- Druid Performance Tuning – worked with Imply.io to implement optimizations for Druid querying and indexing. Benchmarked potential performance optimizations to evaluate cost effectiveness. Achieved over 40% reduction in average report query time.
- Spark Live 2017 Tour – learned Spark background/architecture, SQL interfaces, and had brief demo of streaming and SparkML.
- Completed five part Functional Programming in Scala, improving my knowledge of Scala, functional programming, and Spark.

Technical Skillset

- Design, Architecture, and Fundamentals
  - Strong knowledge of data-structures, algorithms, and computer science fundamentals.
  - Good understanding of advanced functional-programming concepts such as type-classes and monads. Experienced with using a functional programming style in a production codebase including using best practices such as properties-based testing and passing errors/exceptions as values.
  - Extensive experience building REST API’s that follow HTTP best practices such as setting the correct status code in response to error conditions.
  - Experience and understanding of tools and concepts for achieving *concurrency* and *parallelism* such as *asynchronous programming* with futures/promises, event-driven programming, Mapreduce, and atomic primitives.
- Hadoop, Spark and other Big Data tools
  - Extensive experience building data pipelines using Apache Spark with Spark SQL and Dataset API.
  - Experience using both an in-house Mapreduce cluster, on MaprR, and deploying to AWS EMR for faster deployment and scaling.
  - Used Protobuff and Avro to compress logfiles and achieve high compression ratio compared to naïve uncompressed storage
  - Experience maintaining, and tuning Hadoop jobs to solve out of memory issues. Used *pepperdata* to *profile* and *optimize* Hadoop jobs to reduce load on the cluster.
- Web-browser application development
  - Experience with ES6 – using language features such as *lambda expression*, *es6 modules* and *iterators* – with babel for *transcompilation* and integration with *eslint*.
  - Migrated a legacy application **build system** to modern build system based on open source tools such as *gulp*, *npm*, *webpack* and integration with *Github*, and *Jenkins*. Wrote unit tests using *jasmine* and *jest* and integrated in CI builds.
  - Experience with a several open source frameworks including *React*, *Angular 1*, and *Backbone*.
  - Used a pragmatic selection of application widgets and components including *datatables*, *daterangepicker*, *selectize*, *highcharts*, and *d3-sankey*.
  - Used a variety of javascript libraries including *moment*, *moment-timezone*, *underscore*, *immutable*, and *jquery*.
  - Integrated application components with *Mixpanel* to gain insights into *user behavior*.
- Restful Server/HTTP API Development
  - Experience with several frameworks including *vertx.x*, *Springframework*, *Django*, and *Google App Engine*
  - Extensive experience with different varieties of ORMs including Active-Record-style *Django Models* and Data-Mapper-style *Java Hibernate*
  - Experience developing asynchronous/nonblocking and stateless services using vertx.x and expressing callbacks as Java *CompletableFuture*
  - Used Memcache to cache both long-running database queries and API calls to REST services.
  - Experience integrating with external REST API’s including OAuth-authenticated endpoints by *Paypal*, *Facebook*, and *Twitter*.
- Extensive knowledge of Java language, common libraries and JVM
  - Used both sbt and maven to build shaded jars. Used *Jenkins* for *continuous integration* with unit tests using *junit*, *testng*, *spymemcached*, and *mockito.*
  - Used common libraries including *lombock*, *guava*, *fuge*, *jackson*, *joda datetime*, *quartz scheduler,* and *fuge.*
- Unix shell scripting
  - Knowledge of best practices in areas such as process management, building rpms, logging, and monitoring.
  - Experience troubleshooting production issues with common tools such as *htop*, *ps*, *du*, *tail*, etc.
  - Big fan of using simple bash utilities such as *mysqldump*, *mysql*, *awk*, and *crontab* when possible.
- Used many languages in production including *Scala*, *Python*, *Java*, *Javascript*, *PHP*, *Bash*, *and SQL.*
- Experience with many datastores including *MySQL* (for both OLAP and OLTP), *Druid* (for OLAP), and *MongoDB (*for *OLAP)*
- Strong conceptual understanding of *relational databases* and extensive experience with *MySQL*
  - Experience with *data-modeling/table-design* for both *OLAP* and *OLTP* workloads.
  - Strong understanding and appreciation of the guarantees relational databases bring such as *ACID properties*.
  - Experience troubleshooting performance issues with highly complex application-generated OLAP queries.

Relevant Experience and Employment

Software Development Engineer – *The Rubicon Project* – Playa Vista, CA July 2014 – Present

- Wrote new Spark job for reporting on Rubicon's "Prebid" traffic. Used Amazon EMR for rapid deployment and scaling.
- Developed Spark job for "First-Party" dataset ingesting data into Druid to sunset an older reporting product and add new functionality.
- Migrated several legacy Hadoop jobs to Spark using Spark SQL and Dataset API, increasing performance and maintainability.
- Contributed to data-pipeline, API, and application layer for next-generation performance reporting product based on Druid capable of ingesting over a billion aggregated rows per month and allowing an arbitrary selection of over 40 dimensions and over 30 base and derived metrics.
  - Designed new tables, built REST API, and developed application for managing *saved*, *standard*, and *scheduled* reports.
  - Developed new dimension columns such as *Site Groups* that represent many-to-many relationships to deprecate legacy reporting.
  - Worked with *Imply.io* to improve Druid performance. Worked with DevOps engineers to upgrade Druid to the next version.
- Contributed to *Blocked Advertiser Revenue* and *Optimized Price Floors* which produce recommendations based on historical data.
- Developed ETL to ingest financial data into MySQL from a new buy-side product. Worked with QA to ensure data accuracy.
- Performed maintenance and performance optimization on a legacy OLAP reporting product based on *MySQL*.
- Helped build new API and application for managing "seats" which represent the buyer/customer on Rubicon's exchange.

Student – *HMC* Clinic Program with *The Rubicon Project – Respectful Cross Device Attribution* *Sept 2013 – May 2014*

- Worked with three other students, an academic advisor, and a liaison from *The Rubicon Project* to gain industry experience and explore novel approaches to tackling a big-data problem.
- Project's goal is to be able to match devices – to be able to say these two devices might belong to the same person.
- Played a key role in creating the design for the project, pushed for similarity hashing as a technique to group device profiles.
- Researched, designed, and implemented similarity measures that operate on pairs of device profiles.

Software Engineering Intern – Pagewoo/Nearwoo (now defunct) June 2013 – August 2013

- Developed a module to automatically generate information about a user's business. Performed integration with the yelp business search API as well as building a scraping component to find, among other things, relevant pictures that would be useful in banner ad creation.
- Performed other miscellaneous scraping and web development tasks.

Technical Assistant – *Pomona College*; CS052: Foundations of Computer Science Aug 2012 – Dec 2012

- Responsibilities included grading homework and holding office hours to answer questions on homework problems.

Education

Bachelor of Arts, *Computer Science*, *Minor in Mathematics*, *2014*, **Pomona College**, Claremont, California

Personal Projects

- Interactive Tic-Tac-Toe: can you beat the AI?
- Self Learning Meta Tic-Tac-Toe: watch the AI improve its state-evaluation constants using temporal difference learning.
- Interactive Meta Tic-Tac-Toe: test your abilities against the AI. Adjust its search depth parameter to change its skill level. Adjust its evaluation constants to change its behavior.
- Using the Copy Model for Edge Attachment to Evaluate the Stability of Graph Clustering: reproducing and extending results from a paper in graph/social-network analysis.
- E-Flashcards: create, organize, and review digital flashcards.

Career Lessons and Design Philosophy

- Whenever possible, reduce amount of state the program must manage. *"It is better to have 100 functions operate on one data structure than 10 functions on 10 data structures"* – Alan Perlis.
- Make architectural design choices extremely carefully and use the right tool for the job! The code can change, but the data and architecture is much harder to change.
- YAGNI – don't invent abstractions until they are really needed. Borrow abstractions if at all possible.
- Don't try to conquer hard problems or invent abstractions, tools or libraries. Instead borrow from open source and academia.
- Gain inspiration from the philosophy behind great software projects. For example – *Unix philosophy*, *Zen of Python*

Key Abilities, Responsibilities, and Values

Domain	Ability/Value
Design	**Lateral Thinking** – Work with product and key stakeholders to identify, better understand, and design solutions to ***wicked problems*** by understanding (and challenging) *key assumptions*, *identifying unifying first principles*, and ***reframing the problem statement***. **Innovation** – Quickly implement a *proof of concept* or *prototype* to evaluate the feasibility of an idea. **Rigor** – ***Borrow* Abstractions** to achieve simplicity. Follow best practices. Understand and analyze problems through the appropriate academic lens to reduce risk, and avoid pitfalls and achieve *correctness* in areas such as *concurrency, scalability, and reliability*.
Develop	**Continuous Learning** – Quickly learn new code bases, **frameworks**, **tools**, **algorithms**, and **libraries**. **Function Programming** – Use a *pragmatic* functional programming style to improve maintenance costs, speed development, and reduce time spent debugging. **Open Source** – Achieve simplicity, minimize development and **maintenance costs**, and avoid reinventing the wheel by finding, using, and contributing to open source tools, platforms, and libraries.
Debug	**Spidey Sense** – Proactively **anticipate** how changes in one component might cause breaks in a connected or downstream component. **Firefighter** – Identify root causes and rapidly implement solutions to critical *production fires* and bugs. **Rabbit Holing** – Track extremely challenging **bugs** that originate in open source libraries, misconfigurations/problems in production environment, or issues caused by other teams.
Teamwork	**Documentation** – Achieve scalable knowledge sharing by rigorously documenting high-level *design*, *algorithms*, *data-flow*, and *usage*. **Agreeable** – *Assume good faith*. ***Consensus*** is forged by a ***unifying proposal*** that takes into account *competing perspectives*. **Truth** – Avoid *bias*. Resist *bullshit*. *Focus* on what's really important. Qualify one's *level of certainty*. *Differentiate* among factual statements, personal preference, and conjecture. Do the additional research to be *really certain* when answering important questions.