The Data-Driven Future of Biotechnology

Aaron Kimball – CTO

zymergen
About me

• Career goal: Find interesting problems where big data can lend an advantage and build solutions.

• Previously: cloudera, sqoop, WIBI

• CTO of Zymergen since May 2014
A business built on positive feedback loops

Zymergen is a software and data company revolutionizing bioengineering with a combination of machine learning, automation, and data-driven process improvement and search techniques.

... And we use this to optimize the genomes of microbes for chemical production.
Roadmap

- Background: Industrial fermentation
- DNA optimization: the Zymergen challenge
- Software system description and architecture
- Software throughout the Zymergen process
- Conclusions
Roadmap

• Background: Industrial fermentation
• DNA optimization: the Zymergen challenge
• Software system description and architecture
• Software throughout the Zymergen process
• Conclusions
Zymergen: Industrially Fermented Chemicals

Microbes can produce all sorts of carbon-based products:
- Commodity chemicals
- Specialty chemicals
- Agricultural products
- Amino acids
- Flavors and fragrances
- Therapeutics...
Bioengineering in: Industrial Fermentation

Before

After

Source: SigmaAldrich
... in Medicine

1. Extract lymphocytes from the patient’s blood.
2. Infect the lymphocytes with the virus.
3. Inject the lymphocytes with the recombinant gene into the patient.
4. Virus with inserted normal gene.

Penicillin VK Tablets

B.P. 250mg
... in Biological Computation

Electronics
Reset–set latch

Switch

Biology
Bacteriophage λ lysis–lysogeny switch

Synthetic biology
Genetic toggle switch

Heat

IPTG

LC oscillator circuit

C

L

Oscillator

Cyanobacteria circadian oscillator

KaiA

KaiB

KaiC

KaiB

KaiC

Repressilator

P

tetR

lacI

cl

Nature Reviews | Genetics

Image: Nature (“Synthetic biology: applications come of age”)
DNA Construction is becoming routine

"Time for New DNA Synthesis and Sequencing Cost Curves", Rob Carlson
DNA Optimization: A hard problem

• Where do you make changes for optimization, or why?
• To get DNA optimization right, you need to do two things:
  • Do a better job of modeling than we can do as humans
  • Test changes a lot more efficiently.

efficiency = high throughput + high precision
Roadmap

• Background: Industrial fermentation
• DNA optimization: the Zymergen challenge
• Software system description and architecture
• Software throughout the Zymergen process
• Conclusions
Zymergen’s Optimization Workflow

**DESIGN**
New DNA constructs

**BUILD**
Microbes with new DNA

**TEST**
Evaluate microbe performance

**ANALYZE**
Incorporate test results into next design phase
Optimization requires parallel testing
Automation raises the bar
Automation brings analytic challenges

• Propose genomes to test in high volume
• Track workflows at high throughput
• Correlate results with these genomes, and use this in a feedback loop

*High throughput automation requires
a new approach to lab management software*
More and better data on processes & results
- Better predictions
- Better failure analysis
- Data-driven workload determination / optimization
- Greater confidence in methods
- Optimized method runtimes

Higher throughput

Big data, cloud storage

Distributed computation

Open Source ML

Robot automation

“Glass cockpit” for ops

Perpetual Quality Improvement Loop
Roadmap

• Background: Industrial fermentation
• DNA optimization: the Zymergen challenge
• Software system description and architecture
• Software throughout the Zymergen process
• Conclusions
Helix: A software-defined factory

Zymergen’s in-house software for DNA design and analysis

- **DESIGN**: New DNA constructs
- **BUILD**: Microbes with new DNA
- **TEST**: Evaluate microbe performance
- **ANALYZE**: Incorporate test results into next design phase

- Analyze results
- Generate designs, create work orders
- Absorb data
- Guide factory workflows
- Generate designs, create work orders
- Incorporate test results into next design phase
- Evaluate microbe performance
- Analyze results
- Guide factory workflows
Example Helix functions

- Design optimized genomes
- Manage factory workflows
- Quality control analysis
- Integrate multiple data sources
- Calculate microbe performance over prior generations
System architecture diagram

- Research Scientists
- Factory Operations Team
- Lab / Factory robots
- Workflow Tools
- Automation
- LIMS Frontend
- ElasticSearch
- Zymergen REST API
- Job queue
- Big data processing
- Data science
- S3
- LIMS data store
- Analytic databases
Roadmap

• Background: Industrial fermentation
• DNA optimization: the Zymergen challenge
• Software system description and architecture
  • Software throughout the Zymergen process
• Conclusions
Design
New DNA constructs

Build
Microbes with new DNA

Test
Evaluate microbe performance

Analyze
Incorporate test results into next design phase

Analyze results
Guide factory workflows

Absorb data
Generate designs, create work orders

Guide factory workflows
Create work orders
hostStrain = dnaForStrain("e-coli-461")
promoters = load("promoter-lib-2-13-2015.gb")
genes = locateGenes(hostStrain, "YFG*")
create replacePromoter[x](genes, promoters)
Language Design Goals

- Be useful for high-level (gene replacement) and low level (nucleotide editing) work on sets of genomes
- Generate data structures we can use to schedule factory work
- Easy learning curve for scientists; concise scripts
- Scripts should always terminate
- Scripts should be transactional
Implementation

• Written as a Scala DSL
• Genomes and edits are described in Avro data structures
• Uses Jersey JAX-RS client to interface with our persistence REST API
• BioJava library provides sequence, file manipulation functions
• Users submit scripts through a web-based form
DNA is annotated

**Promoter:** p1

**Gene:** YFG1

**Author:** Bob Smith

**Terminator:** t1

**Synonyms:** YFP

**Type:** name

**Name:** YFG1
Genome editing functions

Treating DNA like a string (“ATGACGTAGTCA...”):
- concat()
- delete()
- insertUpstream(), insertDownstream()
- replace()

Genome-aware edit operations:
- replacePromoter() – Replaces nearest promoter attached to an annotated region, or insert if no existing promoter.
Promoter swapping

Promoter swapping is performed by the function `replacePromoter[x](locateGenes(hostStrain, "YFG"), promoters)`.
DNA Data Model

• Codon scripts generate data structures describing changes
• Stored using Avro data structures
  • Rationale: strong nested record support, maps, experience
  • Writer schema stored alongside blob for each object
• Individual objects are immutable – write once, read many
• Two primary datatypes
  • DnaComponent – Stores DNA and its annotations
  • DnaSpecification – Describes an operation on DNA
• Compatible with SBOL, Genbank, and other DNA storage formats

Credits to Barbara Frewen, colleague who designed this
DnaComponent

- Stores a DNA sequence
  - Directly encoded as a string, or as a diff that references another DnaComponent
- Holds common metadata like name, description, source...
- Identified by universal id
- Stores annotations on the genome
  - Identify extent of the annotation, as well as its name and type
  - Validates annotation type exists in our sequence ontology
  - Additional properties may be associated with the annotations
- Stores arbitrary properties; e.g. id of similar DNA strings, aliases...
DnaSpecification

• Describes an operation (**replace**, **subsequence**, **concat**...) on sets of DNA operands
  • Uses lists to describe inputs for each argument (e.g., left & right sides of **concat**)
• (**List x List**) → **List** functions can be applied in either Dot- or Cross-product form.
• Additional string-based arguments & properties permitted
  • Suggested lab protocol to use to perform the operation
  • Metadata templates to attach to DnaComponent outputs
• Also identified by common universal id
• Recursive: Can operate on DnaComponents directly, or other DnaSpecification outputs
Tracking a factory

• Provide validated key metadata across all factory operations
• Make data types extensible by users
• Create a unified namespace for all factory data elements
  • Share data extensions across a variety of data models
  • Example: Quality data can be applied to strains, samples, plates, wells...
LIMS data drilldown
Extensible data models

<table>
<thead>
<tr>
<th>ZId</th>
<th>Model type</th>
<th>Core metadata</th>
<th>Extended attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000004800</td>
<td>“plate”</td>
<td>{ “wellCount”: 96, …  }</td>
<td>Parent id: 20000004600</td>
</tr>
</tbody>
</table>

- **Unified key space**
- **Serialization info**
- **Core record info**
  - Some common to all models; some specific
- **Extended attributes**
  - Dilution factor: 40x
  - Total volume: 1mL

Namespace has dedicated managers outside of eng team.
Example: Plates

• Basic unit of high-throughput laboratory manipulation
• (Typically) a collection of 96 wells in an 8x12 grid
• Metadata:
  • ZId number
  • Name, description
  • Project
  • Hardware type
  • Number of wells (48, 96, 384...)
  • Upstream/downstream plate in workflow
  • ...

Photo: J. Tyson McDonald -- http://jtyson.bol.ucla.edu/
Plates : Barcodes :: Peanut Butter : Jelly

Zld number in 128-barcode format verified by robot workers

Human-readable Zld number

Plate description

Photo: Zld barcode from 96-well plate
Data Discovery

• User queries through ElasticSearch
• Updates sent from REST APIs (backend)
  • Persist to primary data store, then enqueue for search
• Admin interface can trigger full index rebuild
• ElasticSearch *dynamic mapping* used to add user attributes to schema
DESIGN
New DNA constructs

BUILD
Microbes with new DNA

TEST
Evaluate microbe performance

ANALYZE
Incorporate test results into next design phase

Generate designs, create work orders

Guide factory workflows

Absorb data

Generate designs, create work orders

Incorporate test results into next design phase

Analyze results
Result analysis
DNA Sequence Quality Control

Final DNA assembly

Sanger DNA sequencing

Analyze: Sequence matches expectations?

No
Reject construct Analyze failure mode

Yes
Continue to build new microbe

Example Sanger sequencing output data

GAT AAAT CT GGT TTT ATT TCC

cutoff?
QC Score data

• Pass/fail and score associated with samples in database
• Parameters, reason for decision and other metadata stored in JSON blobs
• Users can manually relabel samples to correct workflow, improve future ML training

QC result shows up in context of other data.
DESIGN New DNA constructs
BUILD Microbes with new DNA
TEST Evaluate microbe performance
ANALYZE Incorporate test results into next design phase
Analyze results
Guide factory workflows
Absorb data
Generate designs, create work orders
Incorporate test results into next design phase
Create work orders
Data integration challenges

- Genome specifications
- Image data from colony growth
- Several measuring instruments/robots
- Plate content tracking
- Operational logs from robots
- Sensors...
Performance improvement analysis
Conclusions

• Automated systems and the Internet of Things are changing biology
• Big data + ML + automation = efficiency + flexibility

• Your data model needs to be as or more flexible than the underlying process
  • Avro provides rich data modeling
  • ElasticSearch provides discoverability
  • Users can extend data model without eng team help

• Allow your users to override data by hand; clean as you work
Thanks for listening!

Aaron Kimball
aaron@zymergen.com

PS, we’re hiring...
zymergen.com/careers.php