

Managing and Analyzing Big-Data in Genomics

Sebastien Mondet¹, Ashish Agarwal^{1,2},

Paul Scheid¹, Aviv Madar¹, Richard Bonneau^{1,2},

Jane Carlton¹, Kristin C. Gunsalus¹

1 Center for Genomics and Systems Biology, Dept of Biology, New York University

2 Courant Institute of Mathematical Sciences, New York University

IBM Programming Languages Day 2012

Outline

- 1 Biology, Genetics, Big Data™, HPC
- 2 A DSL Approach To The Software Stack
- 3 OCaml, Ocsigen, ... Experience Report

Introduction

Genomics & Sequencing

Let's over-simplify:

```

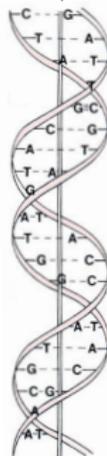
type sample (** Some extract of formerly-living material *)
type library (** Carefully prepared bunch of
                short DNA fragments *)

val lab_tech:
  sample -> protocol -> library real_world_monad

type base = A | C | G | T

type read = base list
  (** one read < one previous DNA fragment *)

val sequencing:
  library -> reagents -> (read list) real_world_monad
  
```



Introduction

Genomics & Sequencing

How it looks like once on the computational side:

```
@SEQ 1
AATAGTAAATCCATTTGTTCAACTCACAGTTTGATTTGGGGTTCAAAGCAGTATCGATCA
+
!' '*((( (**+))%%%++) (%%%) .1***-+*' '))**55CCF>>>>>CCCCCCC65
```

```
@SEQ 2
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
(>CCC (**!>>CCC ' '*(( *%%) .1**+) )%%' '))**5+) (5CC%+%* -+*F>>>C65
```

c.f. [wikipedia:FASTQ_format](https://en.wikipedia.org/wiki/FASTQ_format).

Introduction

Bioinformatics

Then, a branch of computer science takes over:

Alignment [BWA09], [Bowtie09]

```
List.map reads ~f:(fun r -> String.clever_find genome r)
```

Assembly [Assembly10]

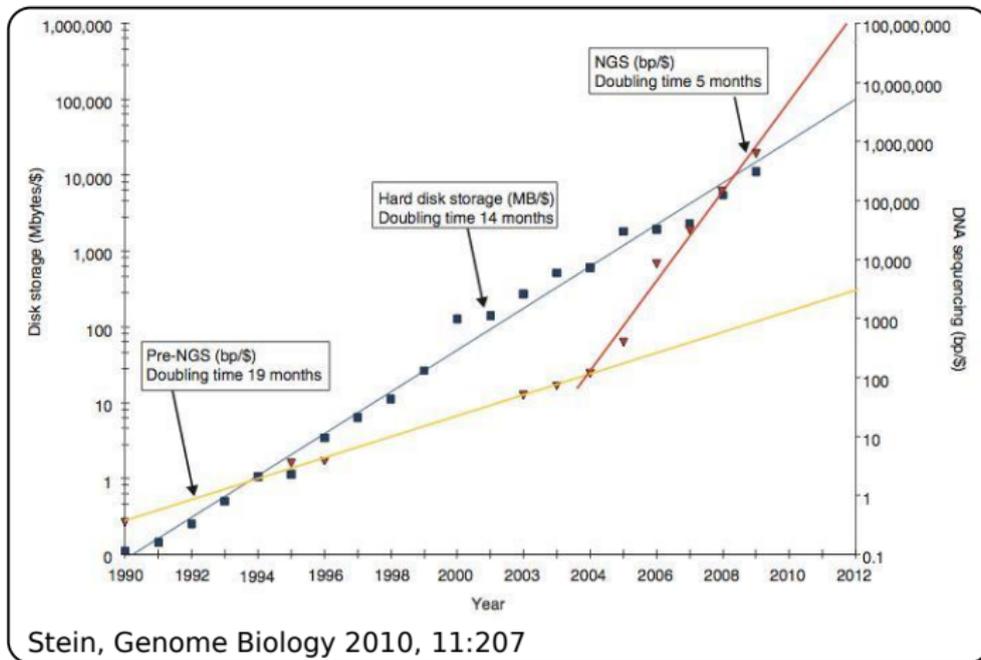
```
val assemble: read list -> genome
```

Annotation [Annot09]

Et cetera.

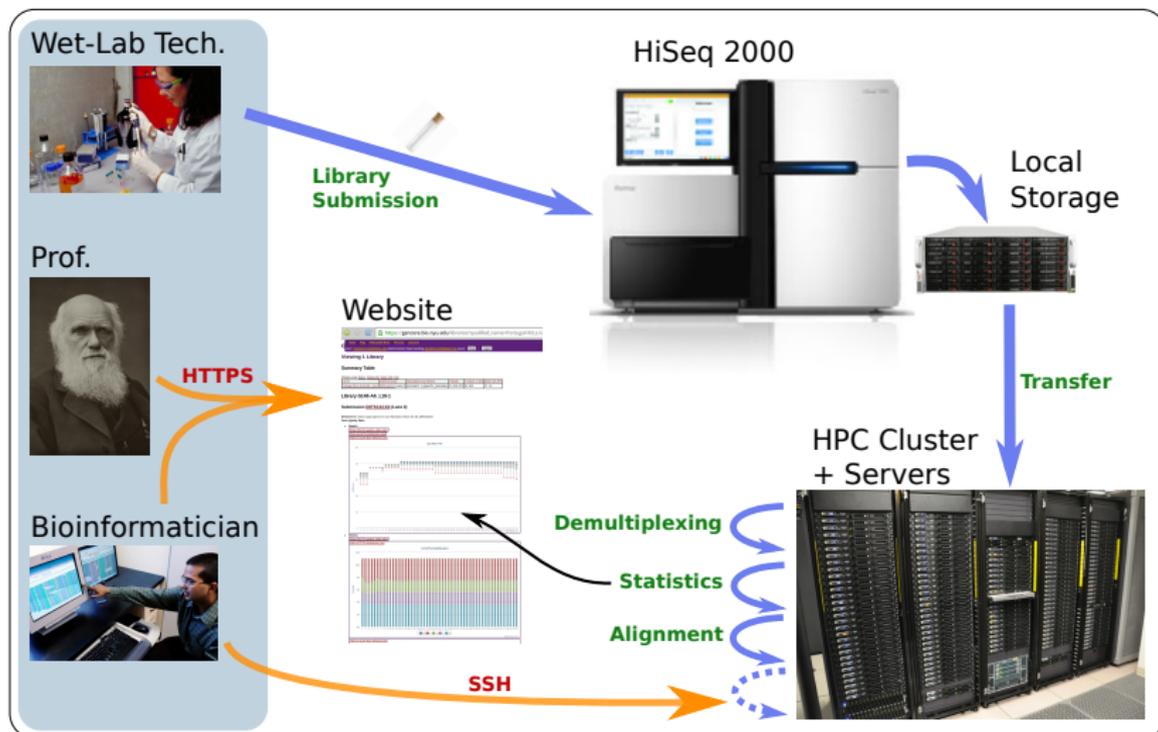
Introduction

Numbers — Moore’s Law of Biology



Introduction

Our Process



The “Layout” DSL

Why?

Need a persistence layer:

- Database → meta-data.
- File Structure → big data.

Need a more high-level representation:

- Virtual file-system: *just* a cache.
- Ensure coherency: DB tables Vs OCaml code Vs File-system.
- SQL is not portable, weakly typed, verbose.
- Quick and safe migrations.

The “Layout” DSL

The DSL Approach

Define a Domain Specific Language:

- “Our” concepts: functions, records, ...
- Better typing: enumerations, non-null by default, file-system elements ...
- Manageable size.

The “Layout” DSL

The DSL Approach

Then, from one single descriptive file:

- Generate the “right” SQL queries & File-system paths.
- Generate well typed OCaml code for accessing the data.
- Generate pretty graphs.
- Handle migrations peacefully.
- Provide an *introspection-like* API.
- Generate safety and consistency checking functions.
- Well-formed backups.

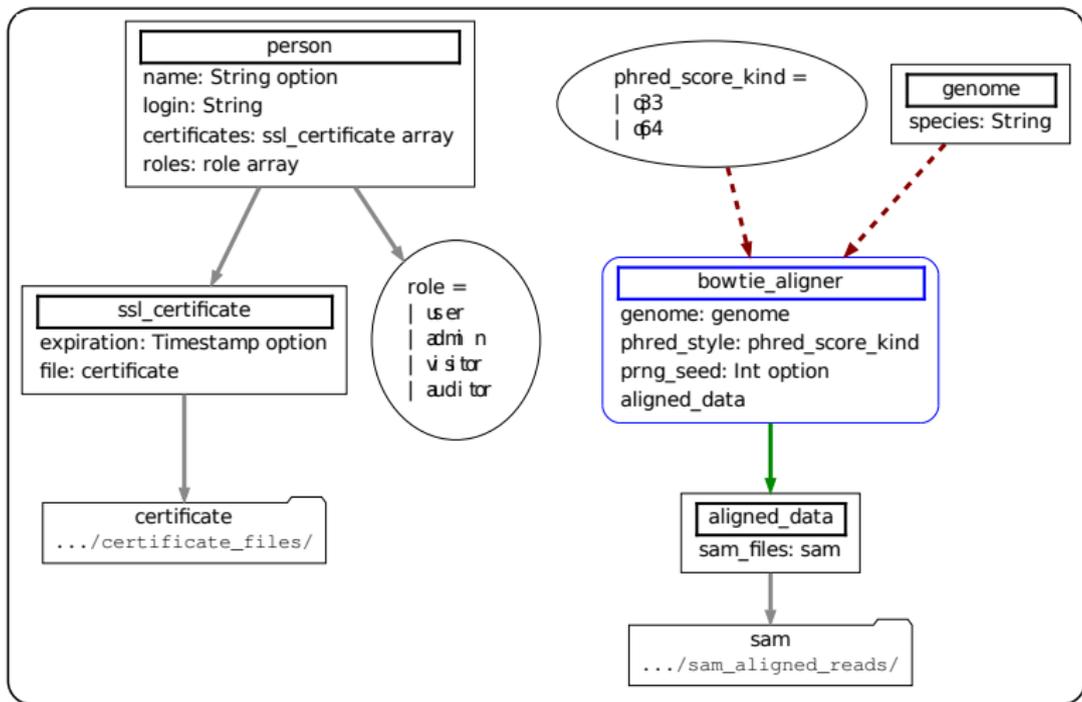
The “Layout” DSL

An Example — The Source

```
(volume certificate certificate_files)
(record ssl_certificate
  (expiration timestamp option)
  (file certificate))
(enumeration role user admin visitor auditor)
(record person
  (name string option)
  (login string)
  (certificates ssl_certificate array)
  (roles role array))
(* ... *)
(function aligned_data bowtie_aligner
  (genome genome)
  (phred_style phred_score_kind)
  (prng_seed int option))
```

The “Layout” DSL

An Example — The Pretty Graph



The “Layout” DSL

An Example — Generated Code

```
module Enumeration_role = struct
  (** The type of {i role} items. *)
  type t = ['user | 'admin | 'visitor | 'auditor] with sexp
  let to_string : t -> string = function
  | 'user -> "user"
  | 'admin -> "admin"
  | 'visitor -> "visitor"
  | 'auditor -> "auditor"
  let of_string_exn: string -> t = function
  (* ... *)

  let of_string s =
    try Ok (of_string_exn s) with e -> Error s
```

The “Layout” DSL

An Example — Generated Code

```
module Record_person = struct
type pointer = { id: int} with sexp
type value = {
  name : string option;
  login : string;
  certificates : Record_ssl_certificate.pointer array;
  roles : Enumeration_role.t array} with sexp
type t = {
  g_id : int;
  g_created : Timestamp.t;
  g_last_modified : Timestamp.t;
  g_value: value} with sexp
```

The “Layout” DSL

An Example — Generated Code

```
module Function_bowtie_aligner = struct
type pointer = { id: int} with sexp
type evaluation = {
  genome : Record_genome.pointer;
  phred_style : Enumeration_phred_score_kind.t;
  prng_seed : int option} with sexp
type t = {
  g_id : int;
  g_result : Record_aligned_data.pointer option;
  g_inserted : Timestamp.t;
  g_started : Timestamp.t option;
  g_completed : Timestamp.t option;
  g_status : Enumeration_process_status.t;
  g_evaluation: evaluation} with sexp
```

The “Layout” DSL

An Example — Generated Code

```
val add_value: ?expiration: Timestamp.t -> file:File_system.
  pointer ->
  (Record_ssl_certificate.pointer,
   [> 'Layout of Layout.error_location
    * Layout.error_cause ]) Flow.t

let add_value ?expiration ~file ~dbh =
  let v = { expiration; file; } in
  let work_m =
    let query =
      Sql_query.add_value_sexp ~record_name:"ssl_certificate"
        (Record_ssl_certificate.sexp_of_value v) in
    Backend.query ~dbh query
  >>| Sql_query.single_id_of_result
  >>= (function Some id -> return {id} |
      None -> error ('wrong_add_value))
  in
  bind_on_error work_m (fun e ->
    error ('Layout (('Record "ssl_certificate": error_location),
                  (e : error_cause))))
```

The “Layout” DSL

An Example — Generated Code

```
module Bowtie_aligner = struct
  let add_evaluation ~genome ~phred_style ?prng_seed ~dbh =
    (* ... *)
  let get ~dbh pointer =
    (* ... *)
  let get_all ~dbh =
    (* ... *)
  let set_started ~dbh p =
    (* ... *)
  let set_failed ~dbh p =
    (* ... *)
  let set_succeeded ~dbh ~result p =
    (* ... *)
```

The “Layout” DSL

An Example — Migrations & Backups

Write

```
val migrator: S-Expression -> S-Expression
```

```
$ hitscore dump-to-file backup_v42
```

```
$ ./migrator backup_v42 backup_v43
```

```
$ hitscore wipe-out-database
```

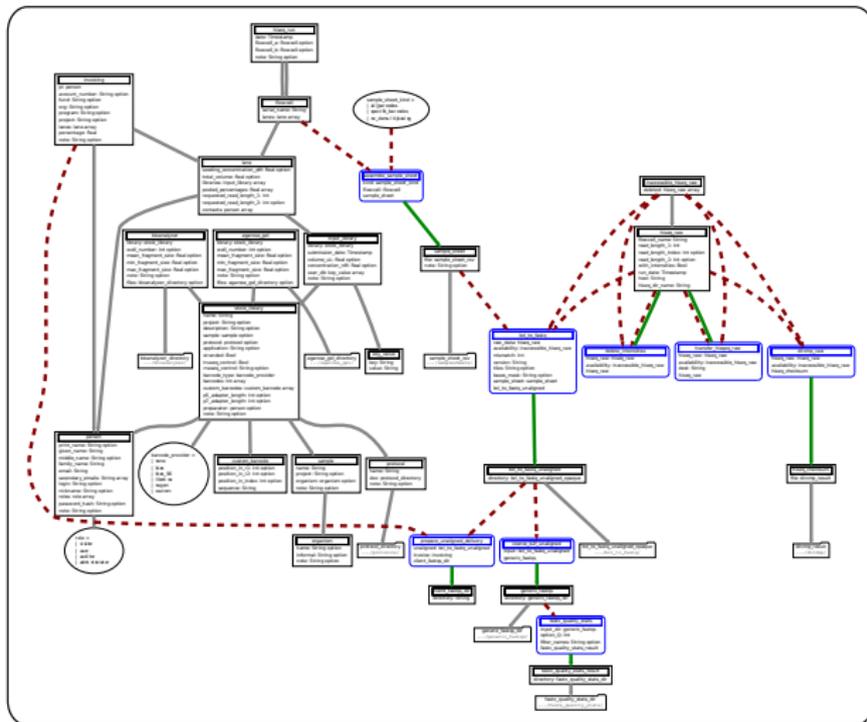
```
$ hitscore init-database
```

```
$ hitscore load-file backup_v43
```

```
$ hitscore verify-layout
```

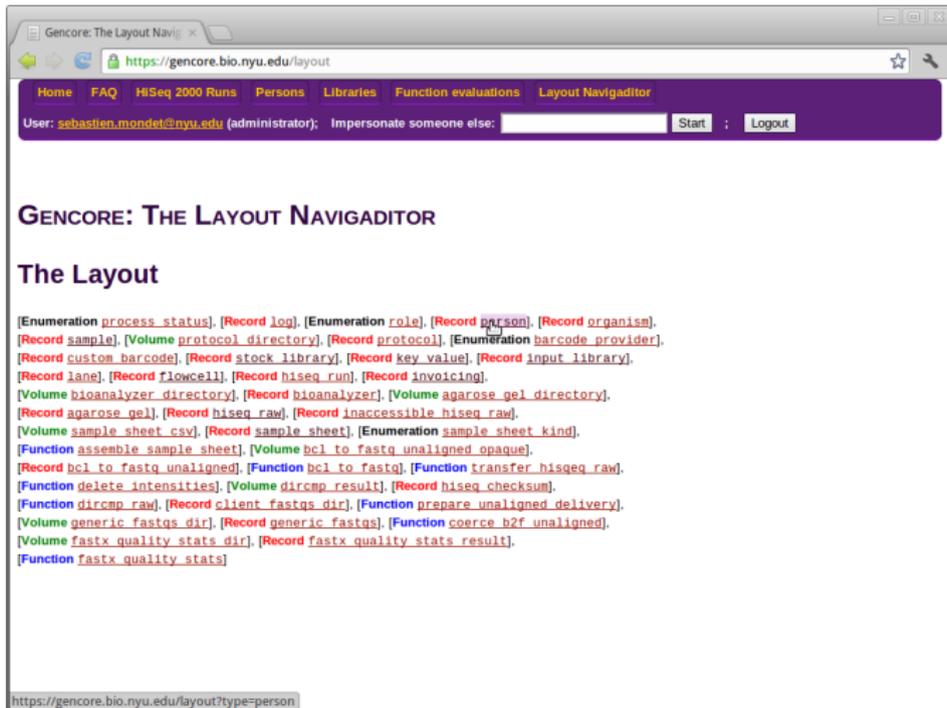
The “Layout” DSL

Our Current Layout



The “Layout” DSL

Introspection-like API



The screenshot shows a web browser window with the URL `https://gencore.bio.nyu.edu/layout`. The page has a purple header with navigation links: Home, FAQ, HiSeq 2000 Runs, Persons, Libraries, Function evaluations, and Layout Navigaditor. Below the header, there is a user login area showing the user `sebastien.mondet@nyu.edu` as an administrator, with fields for impersonating someone else and buttons for Start and Logout.

GENCORE: THE LAYOUT NAVIGADITOR

The Layout

[Enumeration process_status], [Record log], [Enumeration role], [Record person], [Record organism], [Record sample], [Volume protocol_directory], [Record protocol], [Enumeration barcode_provider], [Record custom_barcode], [Record stock_library], [Record key_value], [Record input_library], [Record lane], [Record flowcell], [Record hiseq_run], [Record invoicing], [Volume bioanalyzer_directory], [Record bioanalyzer], [Volume agarose_gel_directory], [Record agarose_gel], [Record hiseq_raw], [Record inaccessible_hiseq_raw], [Volume sample_sheet_csv], [Record sample_sheet], [Enumeration sample_sheet_kind], [Function assemble_sample_sheet], [Volume bcl_to_fastq_unaligned_opaque], [Record bcl_to_fastq_unaligned], [Function bcl_to_fastq], [Function transfer_hiseq_raw], [Function delete_intensities], [Volume dircmp_result], [Record hiseq_checksum], [Function dircmp_raw], [Record client_fastqs_dir], [Function prepare_unaligned_delivery], [Volume generic_fastqs_dir], [Record generic_fastqs], [Function coerce_b2f_unaligned], [Volume fastx_quality_stats_dir], [Record fastx_quality_stats_result], [Function fastx_quality_stats]

`https://gencore.bio.nyu.edu/layout?type=person`

The “Layout” DSL

Introspection-like API

Gencore: The Layout Navig | x

https://gencore.bio.nyu.edu/layout?value=3179&type=person

Home FAQ HiSeq 2009 Runs Persons Libraries Function evaluations Layout Navigator

User: [sebastien.mondet@nyu.edu](#) (administrator); Impersonate someone else: Start ; Logout

- You may add a new person
- You may modify this person

g_id: <i>Identifier</i>	3179
g_created: <i>Timestamp</i>	2011-12-13
g_last_modified: <i>Timestamp</i>	2012-05-29
S-Exp: <i>String</i>	((print_name ("Sebastien Mondet")) (given_name Sebastien) (middle_name ()) (family_name Mondet) (email sebastien.mondet@nyu.edu) (secondary_emails (seb@mondet.org)) (login (sm4431) (nickname ())) (roles (administrator)) (password_hash ()) (note ()))
print_name: <i>String option</i>	Sebastien Mondet
given_name: <i>String</i>	Sebastien
middle_name: <i>String option</i>	
family_name: <i>String</i>	Mondet
email: <i>String</i>	sebastien.mondet@nyu.edu
secondary_emails: <i>String array</i>	seb@mondet.org
login:	sm4431

The “Layout” DSL

Introspection-like API

The screenshot shows a web browser window with the URL `https://gencore.bio.nyu.edu/layout?value=3179&type=person`. The page has a purple navigation bar with links: Home, FAQ, HiSeq 2009 Runs, Persons, Libraries, Function evaluations, and Layout Navigator. Below the navigation bar, the user is identified as `sebastien.mondet@nyu.edu` (administrator). There are fields for impersonating someone else and buttons for Start and Logout.

The main content area displays a table of user information:

Identifier	3179
g_created: <i>Timestamp</i>	2011-12-13
g_last_modified: <i>Timestamp</i>	2012-05-29
S-Exp: <i>String</i>	(print_name ("Sebastien Mondet") (family_name Mondet) (email (sebastien.mondet.org)) (login (sebastien.mondet@nyu.edu)))
print_name: <i>String option</i>	Sebastien Mondet
given_name: <i>String</i>	Sebastien
middle_name: <i>String option</i>	
family_name: <i>String</i>	Mondet
email: <i>String</i>	sebastien.mondet@nyu.edu
secondary_emails: <i>String array</i>	seb@mondet.org
login:	sebastien.mondet@nyu.edu

Next to the table is an **Actions:** box containing:

- [You may add a new person](#)
- [You may modify this person](#)

Below the actions box is a small table:

g_id: <i>Identifier</i>	3179
-----------------------------------	------

The “Layout” DSL

Introspection-like API

The screenshot shows a web browser window with the URL `https://gencore.bio.nyu.edu/layout?value=5867&type=fastx_quality_stats`. The page has a purple navigation bar with links for Home, FAQ, HiSeq 2000 Runs, Persons, Libraries, Function evaluations, and Layout Navigator. Below the navigation bar, the user is identified as `sebastien.mondet@nyu.edu` (administrator). The main heading is **Function fastx_quality_stats**. Underneath, there are 'Actions' including a link to 'You may add a new fastx_quality_stats'. The central part of the page is a table with the following data:

<code>g_id:</code> <i>Identifier</i>	5867
<code>g_result:</code> <i>fastx_quality_stats_result option</i>	5861
<code>g_inserted:</code> <i>Timestamp</i>	2012-04-20
<code>g_completed:</code> <i>Timestamp option</i>	2012-04-20
<code>g_status:</code> <i>process_status</i>	Succeeded
<code>S-Exp:</code> <i>String</i>	((input_dir ((id 5849))) (option_Q 33) (filter_names ("*.fastq *.fastq.gz")))
<code>input_dir:</code> <i>generic_fastqs</i>	5849
<code>option_Q:</code> <i>Int</i>	33
<code>filter_names:</code> <i>Generic option</i>	("*.fastq *.fastq.gz")

Fields `g_inserted:` and `g_completed:` are circled in green in the original image. The browser's address bar at the bottom shows the URL `https://gencore.bio.nyu.edu/layout?value=5861&type=...`.

The “Layout” DSL

Introspection-like API

gencore: The Layout Navigator

https://gencore.bio.nyu.edu/layout?value=5867&type=fastx_quality_stats

Home FAQ HiSeq 2000 Runs Persons Libraries Function evaluations Layout Navigator

User: [sebastien.mondet@nyu.edu](#) (administrator); Impersonate someone else: Start ; Logout

Function fastx_quality_stats

Actions:

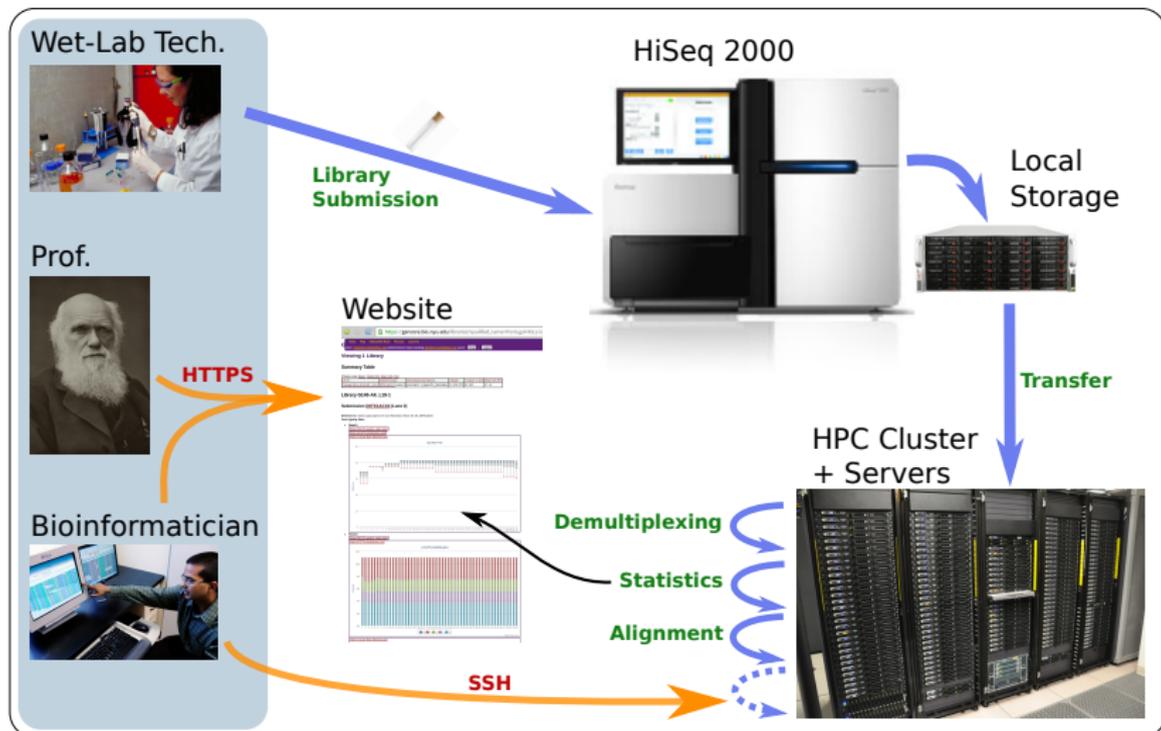
- You may add a new fastx_quality_stats

g_id: Identifier	5867
result: fastx_quality_stats_result option	5861
g_id: Identifier	2012-04-20
g_id: Identifier	2012-04-20
g_result: fastx_quality_stats_result option	5861
S-Exp: String	((input_dir ((id 5849))) (option_Q 33) (filter_names ("*.fastq *.fastq.gz")))
input_dir: generic_fastqs	5849
option_Q: Int	33
filter_names: Generic filter	("*.fastq *.fastq.gz")

https://gencore.bio.nyu.edu/layout?value=5861&type=...

The “Layout” DSL

It's Everywhere



OCaml in Genomics

Experience Report

We use OCaml everywhere with:

- Jane St Core & Batteries
- Lwt
- Ocsigen
- Biocaml
- PG’OCaml, XMLM, Csv, Sqlite, The Cryptokit

OCaml and Asynchronous I/O

Lwt Is Also Everywhere

Writing application-servers & Web-services.

^

Preemptive threads + shared memory + human beings = ☠ ☢ ☣

⇒

Lwt:

- Light-weight threads — Monadic non-preemptive I/O
- Don't block — Don't get preempted
- ocsigen.org/lwt/

OCaml and Asynchronous I/O

Lwt Is Also Everywhere

```
perform_some_complex_io x y z
>>= fun result ->
(* toy "shared mutable state" example: *)
let aux = !global_a in
global_a := !global_b;
global_b := result
return ()
```

OCaml and Asynchronous I/O

Lwt Is Also Everywhere

We actually don't like exceptions:

- Embed a Result/Error monad → *Flow monad*
- Use polymorphic variants for the “error side”
(extensible at will + exhaustive pattern check)

```
module type Flow = sig
  type ('ok, 'err) monad
  val bind : ('ok, 'err) monad -> ('ok -> ('oknext, 'err) monad)
    -> ('oknext, 'err) monad
  val return : 'ok -> ('ok, 'any) monad
  val error : 'err -> ('any, 'err) monad
end
```

OCaml and Asynchronous I/O

An Example — Error Management

```
let f i =  
  match i with  
  | 0 -> return ()  
  | 1 -> error 'its_one  
  | 2 -> error 'its_two  
  | n -> error ('its_a_lot n)
```

```
val f :  
  int -> (unit, [> 'its_a_lot of int | 'its_one | 'its_two ])  
  Flow.t
```

OCaml and Asynchronous I/O

An Example — Error Management

```
bind_on_error (f 42)
  (function
   | 'its_one -> eprintf "One!\n"; return ()
   | 'its_a_lot n -> eprintf "A lot: %d\n"; return ())
```

Characters 51-59:

```
| 'its_one -> eprintf "One!\n"; return ()
  ^^^^^^^^^
```

Error: This pattern matches values of **type** [`< 'its_a_lot of 'a | 'its_one]`

but a pattern was expected which matches values of **type**

```
[> 'its_a_lot of int | 'its_one | 'its_two ]
```

The first variant **type** does not allow tag(s) `'its_two`

OCaml In The Wild

It's Super-Cool

OCaml is not only well-typed:

- Industrial-Strength (core and libraries/frameworks).
- Hackability (Bypass tools, extend build-system).
- Objects and Polymorphic Variants.
- The Future (Coq).

It's being improved on:

- The Marketing.
- The Programmer's Toolkit.

Ocsigen

The Way To Go

A concentrate of awesomeness:

- **Well-Typed** web-programming
⇒ HTML5 **and** services **and** client code !
- `Eliom_output.Caml.register*` ! (⇔ RPC-like programming)
- Choices do not get on your way
DB design, templating, “there is more than one way” ...
- Statically linked native webserver/application.
- `js_of_ocaml`: great but still limited by JS/DOM.
Something is missing there ...

Thanks

Any Questions?

Contacts:

- Ashish Agarwal
ashish.agarwal@nyu.edu
<http://ashishagarwal.org/>
- Sebastien Mondet
sebastien.mondet@nyu.edu
<http://seb.mondet.org>

References I

- [BWA09] Heng Li, and Richard Durbin ; *Fast and accurate short read alignment with Burrows-Wheeler transform*. Bioinformatics Volume 25 Issue 14, 2009.
- [Bowtie09] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg ; *Ultrafast and memory-efficient alignment of short DNA sequences to the human genome*. Genome Biol. Volume 10 Issue 3, 2009.
- [Assembly10] Jason R. Miller, Sergey Koren, and Granger Sutton ; *Assembly algorithms for next-generation sequencing data*. Genomics. 2010 June; 95(6): 315–327, 2010.

References II

- [Annot09] Peter Bakke, Nick Carney, Will DeLoache, Mary Gearing, Kjeld Ingvorsen, Matt Lotz, Jay McNair, Pallavi Penumetcha, Samantha Simpson, Laura Voss, Max Win, Laurie J. Heyer, and A. Malcolm Campbell ; *Evaluation of Three Automated Genome Annotations for Halorhabdus utahensis*. PLoS ONE 4(7): e6291, 2009.

The Flow Monad

Why and What

Cooperative (or not) Threads:

- Be able to use *Lwt*, *Async*, or standard preemptive threading.
⇒ Functor over an I/O monad.

Don't like exceptions:

- Need a **Result Monad** (a.k.a. “error monad”)
Already using a monad ⇒ Monad Transformer

```
module type Result_IO_monad = sig
  type ('ok, 'err) monad
  val bind : ('ok, 'err) monad -> ('ok -> ('oknext, 'err) monad)
    -> ('oknext, 'err) monad
  val return : 'ok -> ('ok, 'any) monad
  val error : 'err -> ('any, 'err) monad
end
```

The “Layout” DSL

An Example — Generated Code

```
with_database configuration (fun ~dbh ->
  let layout = Classy.make dbh in
  layout#library#all
  >>= map_sequential ~f:(fun lib -> lib#preparator#get)
  >>| List.dedup
  >>= map_sequential ~f:(fun prep_person ->
    prep_person#set_roles ('preparator :: prep_person#roles))
  >>= fun _ ->
  return ())
```