Turning Big data into knowledge

Techniques and Tools for Parallel Computing on Online Data Streams in Systems Biology and Epidemiology

Marco Aldinucci
University of Torino, Italy
Outline

• Stochastic Formal System Biology
• From Distributed to Multicore and back
• On programming models
  • FastFlow
• The CWC parallel simulator for sys bio
• Some preliminary results
Formal synthetic system biology
System Biology & Gillespie’s algorithm

- Traditionally studied with continuous ordinary differential equations (ODE)
  - bulk reactions, i.e. average behaviour

- Gillespie algorithm: discrete and stochastic simulation of a system via explicit simulation of each reaction

- Gillespie realization represents a random walk that exactly represents the distribution of the master equation (i.e. ODEs)
  - under some hypothesis
Gillespie’s algorithm [77]

1. **Initialization**: Initialize the number of molecules in the system, reactions constants, and random number generators.

2. **Monte Carlo step**: Generate random numbers to determine the next reaction to occur as well as the time interval. The probability of a given reaction to be chosen is proportional to the number of substrate molecules.

3. **Update**: Increase the time step by the randomly generated time in Step 2. Update the molecule count based on the reaction that occurred.

4. **Iterate**: goto Step 2 unless the number of reactants is zero or the simulation time has been exceeded.
Increasingly popular approach

- Sometime more informative than (ODE)
  - multi-stability, divergent or rare behaviours, peaks, ...
  - multi-scale systems
    - e.g. deriving macro behaviour from micro

- More computational demanding
  - much more, especially in motivating cases
Increasingly popular approach

- Bio-PEPA [Hillston, Ciocchetta]
- SPiM [Cardelli, Phillips]
- Stochastic Pi [Priami]
- Stochkit [Petzold]
- Spatial Pi [Uhrmacher]
- Calculus of Wrapped Components [our own]
  - kinetics: mass-action, Michaelis–Menten, Hill ...
  - ...

The Calculus of Wrapped Compartments (CWC)

A **term** is intended to represent a biological system. A term is built by means of the **compartment** constructor, \((- [\cdot] -)\), from a set \(E\) of **atomic elements**, ranged over by \(a, b, c, d\). A **simple term** is defined as:

\[
t ::= a | (\overline{a} \overline{t})
\]

We write \(\overline{t}\) to denote a (possibly empty) multiset of simple terms \(t_1 \ldots t_n\). Similarly, with \(\overline{a}\) we denote a (possibly empty) multiset of atoms.

Dynamics of SCWC

Rewrite rules are defined as pairs of terms, in which the left term characterizes the portion of the system in which the event modelled by the rule can occur, and the right one describes how that portion of the system is changed by the event.

### Biomolecular Event | Examples of CWC Rewrite Rules
---|---
State change | \(a \mapsto b\)
Complexation | \(a \rightarrow b \mapsto c\)
Catalyzed membrane crossing | \(a (b \cdot x \cdot y) \mapsto (b \cdot x \cdot a \cdot y) \quad (b \cdot x \cdot a \cdot y) \mapsto a (b \cdot x \cdot y)\)

Stochastic Rules

Rules are decorated with a **rate** (speed of the reaction).

A **Stochastic Rewrite Rule**, \(R\), is denoted by \(P \xrightarrow{k} P'\).

The stochastic semantics is given by transitions between terms labeled with the rule applied, \(R\), and a transition rate depending on the rate of rule \(R\):

\[
\overline{t} \xrightarrow{R, k \times p} \overline{t}'
\]

where \(R\) is \(P \xrightarrow{k} P'\), and \(p\) is the number of different ways in which the pattern \(P\) may match \(\overline{t}\) \((\overline{t} = C[P]\sigma)\) and such that \(\overline{t}' = C[P'\sigma]\) for some context \(C\) and variable instantiation \(\sigma\).
Ex: HIV and immune response (progression to AIDS)

- Virus, all phenotypes (V4, V5)
- The spike suggests the mutation V5 → V4 (more aggressive)
- Immune response $Z = Z_4 + Z_5$
- Remain stable (but for the peak).
- Now $Z_4$ decrease and $Z_5$ increase
- I.e. HIV is turning into AIDS more rapidly

• Peaks are informative events,
  • virus mutation triggers AIDS progression
  • hardly detected with ODEs

• high resolution required to detect spikes,
  • each trajectory can be over 6G Bytes of data

• and thousands of trajectories are needed
  • compute everything, save everything, move and join all data, analyse all data, then get first results
  • often to discover parameters are wrong ...
• It is Monte Carlo,
  • well understood
  • easy to parallelise on different trajectories
• it is Monte Carlo w Markov Chains models (CTMC)
  • single trajectory: no parallelisation without relaxation
  • compute time \neq \text{simulation time}
  • compute time for different trajectories heavily unbalanced
    • fast reactions and slow reactions, some not interesting (e.g. water-steam-water)
Unbalancing + filtering

- few trajectories (e.g. the interesting ones) can significantly delay the completion of others
- over-provisioning don’t help that much
- simulated time moves at different pace w.r.t. wall-clock time
- data joining from different trajectories should be aligned at the same simulation time
• It is Monte Carlo,
  • well understood
  • easy to parallelise on different trajectories
• it is Monte Carlo w Markov Chains models (CTMC)
  • single trajectory: no parallelisation without relaxation
  • compute time ≠ simulation time
  • compute time for different trajectories heavily unbalanced
    • fast reactions and slow reactions, some not interesting (e.g. water-steam-water)
• it is Monte Carlo AND data analysis
  • data is big, analysis can be very expensive and it typically starts after the simulation
  • the whole workflow is perceived too “slow” by bio-scientists to be really useful
From Distributed to Multicore and back

In BioSims the whole trajectory is needed, easily GB of data

The whole dataset, TB of data, should be re-read to extract statistical estimators

Model description → Pre-processing e.g. parsing → Scheduler Parameter Sweep

Worker 1 run simulations → Disk 1

Worker n run simulations → Disk n

Collectors Trajectories

Post-processing e.g. parsing → Disk
From Distributed to Multicore and back

Now the issue become a real problem

**Bottlenecks:** disk and memory

**Post-processing:** not pipelined and often not parallel at all
From Distributed to Multicore and back

- Multi Carlo sims for Bio are I/O-bound
  - Sampling reduce I/O traffic but worsen precision and analysis of “strange” dynamics (spikes, diversion from average, etc.), which observation motivates stochastic analysis (ODEs)

- Data analysis is also I/O-bound
  - if approached is a “post-processing” fashion, data should be retrieved from the disks

- The porting of distributed solution “as is” on multicore is going insist on weakness of multicore architectures
  - Memory wall, I/O, disk
  - SIMD/GPGPUs do not change the analysis substantially
From Distributed to Multicore and back

• The same arguments holds on distributed, grids, and clouds as soon as the workflow is considered as a whole
  • simulation, data collection and merging, analysis

• Rationale
  • Manage data as stream, compute online
    • May require more computing and less bandwidth
    • Computation should be designed to be pipelined
  • Establish fast data paths across cores/hosts
  • Avoid low-level concurrency management
    • Portability, performance, portability of performance, maintenance, porting from sequential
Quotes from P. Beckman EuroPar 11 “exascale” keynote

- coarse grain concurrency is nearly exhausted
- it is not about Flops, it is about data movement
- programming systems should be designed to support fast data movement and enforce locality
- shared-memory & inter-socket messaging
- we need a programming model
  - a computer language is not a computing model
  - a library is not a computing model
- we need an efficient and compositional run-time
Shared memory

Beowulf

Grid

MPP

P3L 1991

SKiE 1997

OCamlP3L 1998

Eskimo 2003

ASSIST 2001

ASSISTant 2008

GCM 2008

Lithium 2002

Muskel 2006

FastFlow 2009

GPGPUs

Clouds, clusters of multi-cores and many-cores

Patterns/skeletons & streams

Macro Data Flow

Autonomic

1991

1997

1998

2000

2002

2006

2008
it is not about Flops, it is about data movement, we need compositional run-time

- Streams
  - focus on data movements at the prog model level
  - clear semantics
  - support compositionally and also locality
    - the latter is a bit more counter-intuitive
- High-level programming
  - e.g. patterns
- Patterns + streams
  - can be implemented efficiently on both multi-core, distributed, and both
FastFlow

http://mc-fastflow.sourceforge.net/
FastFlow (multicore)

Applications on multicore, many-core
Efficient and portable - designed with high-level patterns

Streaming network patterns
Skeletons: pipeline, map farm, reduce, D&C, ...

Arbitrary streaming networks
Lock-free SPSC/MPMC queues + FF nodes

Simple streaming networks
Lock-free SPSC queues + threading model

Multicore and manycore
SMP: cc-UMA & cc-NUMA
Layer 1: Simple streaming networks

4 sockets x 8 core x 2 contexts

Xeon E7-4820 @2.0GHz Sandy Bridge
18MB L3 shared cache, 256K L2

MPI is ~190 ns at best
(D.K. Panda)
Layer 1: Simple streaming networks

Fig. 3: Unbounded wait-free uSPSC queue implementation.

EuroPar 2012.
Wed 29 Aug - B3 multicore 14.30-16.00
Layer 1: Simple streaming networks

http://www.1024cores.net/home/technologies/fastflow

4 sockets x 8 core x 2 contexts

Xeon E7-4820 @2.0GHz Sandy Bridge
18MB L3 shared cache, 256K L2

<35 ns irrespectively of the mapping

Linked list + circular buffer
FastFlow queue (our result)

12x faster

Linked list with pooling
Opt Michael-Scott queue (our result)

20x faster

Linked list w/ dyn alloc
Michael-Scott queue
well-known ~ 400 citations

Speedup

n. threads

Throughput (msg/s)

1 8 16 24 32 40 48 56 64

50K 100K 150K 200K

buffer size

nanoseconds

1 8 16 24 32 40 48 56 64

50K 100K 150K 200K 200K 250K 300K 350K

buffer size

nanoseconds

1 8 16 24 32 40 48 56 64

50K 100K 150K 200K 200K 250K 300K 350K

buffer size

nanoseconds

1 8 16 24 32 40 48 56 64

50K 100K 150K 200K 200K 250K 300K 350K

buffer size

nanoseconds

1 8 16 24 32 40 48 56 64

50K 100K 150K 200K 200K 250K 300K 350K

buffer size

nanoseconds

1 8 16 24 32 40 48 56 64

50K 100K 150K 200K 200K 250K 300K 350K

buffer size

nanoseconds
Layer 3: streaming networks patterns

- Composition via C++ template meta-programming
  - CPU: Graph composition
  - GPU: CUDA streams
  - CPU+GPU: offloading
- farm{ pipe }
- pipe(farm, farm)
- pipe(map, reduce)
- ....
Layer 3: streaming networks patterns

- **farm**
  - on CPU - master-worker - parallelism exploitation
  - on GPU - CUDA streams - automatic exploitation of async comm

- **pipeline**
  - on CPU - pipeline
  - on GPU - sequence of kernel calls or global mem synch

- **map**
  - on CPU - master-worker - parallelism exploitation
  - on GPU - CUDA SIMT - parallelism exploitation

- **reduce**
  - on CPU - master-worker - parallelism exploitation
  - on GPU - CUDA SIMT (reduction tree) - parallelism exploitation

- **D&C**
  - on CPU - master-worker with feedback - // exploitation
  - on GPU - working on it, maybe loop+farm
Layer 3: streaming networks patterns (easy to port)
+ distributed

- **Generic** ff_node **is subclassed to** ff_dnode
- ff_dnode **can support network channels**
  - P2P or collective
  - used as frontier node of streaming graph
  - can be used to merge graphs across distributed platforms
- **No changes to programming model**
  - at least require to “add” stub ff_dnode
  - when passing pointers data is serialised
    - serialisation hand-managed (zero-copy, think to Java!)

CWC simulator example
Start new simulations, steer and terminate running simulations.

Wall-clock time

Simulation

Trajectory reduction

1's runs —— 2's runs —— 3's —— 4's ——
reduce reduce reduce reduce

Gain

Generation of simulation tasks

Simulation pipeline

Alignment of trajectories

Raw simulation results

Generation of sliding windows of trajectories

Displacement

Gather

Statistical

Engagement

Display of results

Graphical User Interface

Main pipeline

Start new simulations, steer and terminate running simulations

Incomplete simulation tasks (with load balancing)

Start new simulations, steer and terminate running simulations

Trajectory 1

Trajectory n
simulation-time aligned trajectories
Incomplete simulation tasks (with load balancing) are generated and dispatched for alignment of trajectories. After alignment, sliding windows of trajectories are generated and dispatched for filtered simulation results.

The mean and variance of simulation results are calculated, and a k-means analysis is performed to display the results in a Graphical User Interface. The simulation pipeline starts new simulations, steers and terminates running simulations.
Alignment of trajectories (scatter cons)

Generation of simulation tasks (scatter producer)

Simulation pipeline

Start new simulations, steer and terminate running simulations

Analysis pipeline

Display of results

Graphical User Interface

Serialisation, coalescing

Allocation (zero-copy 0MQ), deserialisation

Simulation pipeline (host 0)

Simulation pipeline (host 1)

Simulation pipeline (host n)

Start new simulations, steer and terminate running simulations
Performance (preliminary)

Intel Nehalem 32-core

Cluster 8x Intel Xeon 6-core Infinband (IPoIB) courtesy of Mellanox
Involved data

- Simple examples (neurospora, ...)
  - 2-4 double * n. of variables * n. of samples x n. of trajectories * cases in sensitivity analysis
  - e.g. 4*8*4*1M*1k*8 ~ 1 TBytes

- HIV 6GB x 1024 trajectories ~ 6TB

- The more observed variables, precision, cases for sensitivity analysis the more data
• **CMS: Compact Muon Solenoid at CERN**
  
  • 3500 scientists, 180 Universities and Research Labs (40 countries)
  • CMS is like a ~75 MegaPixels Digital Camera. 40M “photos” / s
    Selection of 300 ‘photos’ / s ~450 MB / s from the detector are ~PBs
    of data / year
  • CERN has (of course) its well established data flow and
    infrastructure, however ...
**Data flow of the CMS experiment**

**Tier-0**
- **Tier-0** (the accelerator centre)
  - Data acquisition & initial processing
  - Long-term mass data storage
  - CMS CERN Analysis Facility (latency-critical data processing, high priority analysis)
  - Distribution of data to Tier-1 centres

**7 Tier-1s**
- **Tier-1** ("online" to the DAQ)
  - High availability centers
  - Custodial mass storage of share of data
  - Data reconstruction and reprocessing
  - Data skimming
  - Distribute analysis data to Tier-2s

**~50 Tier-2s**
- ~50 Tier-2s in ~20 countries
  - End-user physics analyses
  - Detector Studies
  - MonteCarlo Simulation to Tier-1

**CAF**
- WLCG Computing Grid Infrastructure
Innovative Methods for Particle Colliders at the Terascale
(2012-2015, oversimplified vision)

- Huge data
- Diff (somehow)
- Filtered data to be analysed (much smaller)
- Monte Carlo simulation of “known” traces
- Monte Carlo simulations smaller but frequent requires great “reactivity” of the workflow to tune models
- New speculations
- New knowledge

Particle physicists

Theoretic physicists

Preliminary results on channel $H \rightarrow ZZ \rightarrow 4l$
shown at Higg’s boson claim Jul 2012
(Amapane at al.)
Formalising the cell cycle switch

**Numerical analysis of a comprehensive model of M-phase control in *Xenopus* oocyte extracts and intact embryos**

Bela Novak* and John J. Tyson†

Department of Biology, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24060-0406, USA

*Permanent address: Department of Agricultural Chemical Technology, Technical University of Budapest, 1521 Budapest Gellért Ter 4, Hungary

†Author for correspondence

![Diagram](image)

The diagram illustrates the regulatory interactions between unreplicated DNA, pre MPF, active MPF, and cdc25. The symbols represent the phosphorylation states of these components. For example, unreplicated DNA (+) and (-) indicate the presence and absence of phosphorylation, respectively.

**Note:**

- Positive feedback loops: Active MPF stimulates its own production from tyrosine-phosphorylated dimers by activating Cdc25 and inhibiting Wee1. We suspect that these signals are indirect, but intermediary enzymes are unknown and we ignore them in this paper. The signals from active MPF to Wee1 and Cdc25 generate an autocatalytic instability in the control system. We indicate also an 'external' signal from unreplicated DNA to Wee1 and Cdc25, which can be used to control the efficacy of the positive feedback loops.
- Negative feedback loops: Active MPF stimulates its own production from tyrosine-phosphorylated dimers by activating Cdc25 and inhibiting Wee1. We suspect that these signals are indirect, but intermediary enzymes are unknown and we ignore them in this paper. The signals from active MPF to Wee1 and Cdc25 generate an autocatalytic instability in the control system. We indicate also an 'external' signal from unreplicated DNA to Wee1 and Cdc25, which can be used to control the efficacy of the positive feedback loops.

The letters a, b, e, and f are used to label the rate constants for these reactions in Fig. 2.

**(C) Negative feedback loop:** Active MPF stimulates its own production from tyrosine-phosphorylated dimers by activating Cdc25 and inhibiting Wee1. We suspect that these signals are indirect, but intermediary enzymes are unknown and we ignore them in this paper. The signals from active MPF to Wee1 and Cdc25 generate an autocatalytic instability in the control system. We indicate also an 'external' signal from unreplicated DNA to Wee1 and Cdc25, which can be used to control the efficacy of the positive feedback loops. The letters a, b, e, and f are used to label the rate constants for these reactions in Fig. 2.
Direct competition: unstable switch

- x catalyzes the transformation of y into x
- y catalyzes the transformation of x into y

\[
\begin{align*}
\text{x} & \rightleftharpoons \text{y} \\
\text{x} + \text{y} & \rightarrow \text{x} + \text{x} \\
\text{y} + \text{x} & \rightarrow \text{y} + \text{y}
\end{align*}
\]

- This system is bistable, but
  - Convergence to a stable state is slow (a random walk).
  - Any perturbation of a stable state can initiate a random walk to the other stable state.
  - With 100 molecules of x and y, convergence is quick, but with 10000 molecules, even at the same concentration (adjusting the rate) you will wait for a long time.

CWC syntax

\[
\begin{align*}
\text{T: } & a \rightarrow c & 10 & \rightarrow a \ a \\
\text{T: } & c \rightarrow a & 10 & \rightarrow c \ c
\end{align*}
\]
... after a number of transformations: a stable switch faithfully modelling cell switch

CWC syntax

\[ T : a \xrightarrow{10} c \xrightarrow{10} b \quad T : c \xrightarrow{10} a \xrightarrow{10} b \quad T : b \xrightarrow{10} a \xrightarrow{10} a \quad T : b \xrightarrow{10} c \xrightarrow{10} c \]
The Shishi Odoshi

- A Japanese scarecrow (scare−deer)
  - Used by Bela Novak to illustrate the cell cycle switch.

![Diagram of Shishi Odoshi](image)

http://www.youtube.com/watch?v=Vbvec7rtsc&NR=1&feature=fwp

empty + tap → tap + full
up + full → full + dn
full + dn → dn + empty
dn + empty → empty + up

To make it into a full trammel (dotted line), we could make the up position mechanically open the tap (i.e. take up = tap)

Influx Oscillators

- Similar but:
  - The two−input switches are replaced by one−input switches which are reset by constant influxes.

![Diagram of Influx Oscillators](image)

\[ r=s100, c1g−c2g=4000 \]

Works best with \( s=r \).

Needs constant influx of \( c1,c2 \)

Novak−Tyson Oscillator

- First switch
  - Is the ‘transformed’ AM switch in one−input configuration (driven by constant influx of cyclin).

- Second switch
  - Is a simple two−stage switch working as a delay (the first switch is so good in terms of hysteresis that the second switch is not very critical for oscillation).
  - It can be replaced by a one−stage switch (Frerell’s cell cycle oscillator) but oscillation is a bit harder to obtain.

- Connection
  - Single links, as in the influx oscillator.

The 2AM Limit−Cycle Oscillator

- Two AM switches in a Trammel pattern

![Diagram of Trammel Pattern](image)

The red reactions need to be slower (even slightly) than the black reactions, but otherwise the oscillation is robust. Oscillation stops at 10 vs. 10 and 1 vs. 10. Here the rates are 8 vs 10.0 top, and 2 vs 10, bottom.

(Simple limit−cycle oscillators in the literature have very critical rate ranges.)
• Demo: unstable switch
Schlögl model
autocatalytic, trimolecular reaction
scheme (bistable)

\[ A + 2X \xrightarrow{k_1} 3X, \]
\[ B \xrightarrow{k_3} X. \]

\[ T : a \quad a \xrightarrow{0.03} a \quad a \quad a \quad T : a \quad a \quad a \xrightarrow{0.0001} a \quad a \quad T : b \xrightarrow{200} b \quad a \quad T : a \xrightarrow{3.5} \bullet \]
Bacteriophage λ life cycle integration of a strand of DNA in the molecule of E. coli DNA (multi-stable)
Transcriptional regulation in Neurospora (circadian clock period detection)
Conclusions

- Talk focused on programming model
  - Many important “in a cloud” ignored in the talk: middleware, faults, ...
- Data movement & high-level are key features
  - helps the mapping of features to platforms and performance portability
  - ease the design
  - MapReduce is an instance, should be not the only one

- Formal biology at embryonal stage
  - similar data & computation problems in analysis “in vitro” experiments
  - increasing interest from industrial and “core” bio scientist for parallel computing
Acknowledgements

- **FastFlow**
  - Massimo Torquati (Pisa), Marco Danelutto (Pisa), Peter Kilpatrick (Belfast), Massimiliano Meneghin (IBM Research Dublin)

- **Case studies, biological models**
  - Luca Cardelli (Microsoft), Pietro Liò (Cambridge), Andrea Bracciali (Stirling), Cristina Calcagno (Torino)

- **CWC simulation implementation**
  - Maurizio Drocco (Torino), Fabio Tordini (Torino)

- **CWC language design**
  - Maurizio Drocco, Eva Sciacca, Salvatore Spinella, Mario Coppo, Angelo Troina, Ferruccio Damiani (Torino)

- **Graphical User Interface**
  - ETICA company

- **Infiniband cluster and computing facilities**
  - Mellanox, HPC Advisory Council