A software framework for parallel bioinformatics on computational grids

Hans De Sterck[1], Rob Markel[2,3,4], and Rob Knight[3]

[1]Department of Applied Mathematics, CU Boulder (desterck@colorado.edu)

[2]Department of Interdisciplinary Telecommunications, CU Boulder

[3]Yarus Lab, MCDB, CU Boulder

[4]Scientific Computing Division, NCAR

Bioinformatics Supergroup

Monday, November 3, 2003

Outline

(1) Introduction: why parallel bioinformatics on computational grids

(2) The TaskSpaces framework for grid computing

(3) Application: finding correctly folded active RNA motifs

(4) Conclusions and future work

(1) Introduction: why parallel bioinformatics

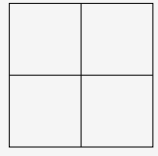
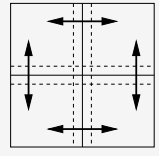
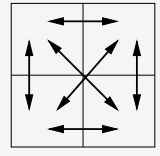
- bioinformatics: large data sets
  - genomics: 28.5 billion basepairs in Genbank (2002 release), 200.000 species
  - proteomics: millions of DTA files per sample

- bioinformatics: computationally expensive algorithms
  - phylogenetics: exponential in number of species
  - proteomics: peptide spectra $\longrightarrow$ proteins $\longrightarrow$ genes

$\Rightarrow$ 1 CPU is not enough, need parallel computing for bioinformatics

parallel bioinformatics

- parallel computing:
  decompose the problem into subproblems (1 per processor)
- classification of problems in terms of communication needs:

no communication	neighbor communication	global communication
		

‘taskfarming’

‘distributed computing’ ‘parallel computing’

$\Rightarrow$ parallel bioinformatics

Introduction: why grid computing?

- where can I find computational power?
  - my desktop machine (1 CPU)
  - other computers in my lab ($\sim$ 10 CPUs)
  - linux cluster at CS department ($\sim$ 100 CPUs)
  - parallel supercomputer at National Center for Supercomputer Applications (Illinois) ($\sim$ 1000 CPUs)
  - other parallel supercomputer at San Diego Supercomputer Center ($\sim$ 1000 CPUs)

  note: it is easy to get accounts on large NSF-funded machines

- 'dream': depending on problem size, I want to be able to run the problem on any of these machines
  (or all at the same time when I have a very large problem, or when I have to beat a deadline!)

why grid computing?

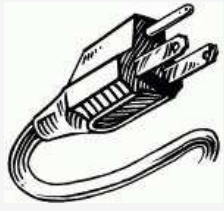
- problems with this ‘dream’:
  - variety of hardware
  - variety of operating systems, OS versions
  - variety of software versions
  - variety of queueing systems
  - need to install and maintain code on all those machines?
  - do I distribute work/data by hand? scripts? how are results centralized?
  - what happens to scripts when machines are added, removed?
  - . . .

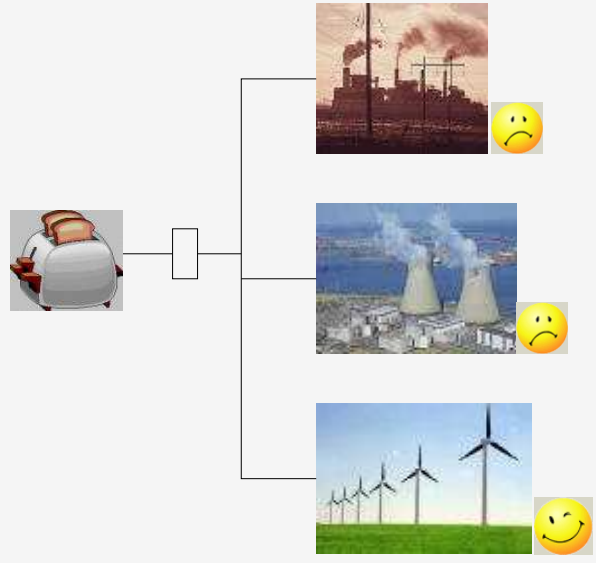
grid computing

- an answer: concept of ‘grid computing’
- analogy: power grid
  - user wants electrical power
  - user doesn’t care where the power is produced

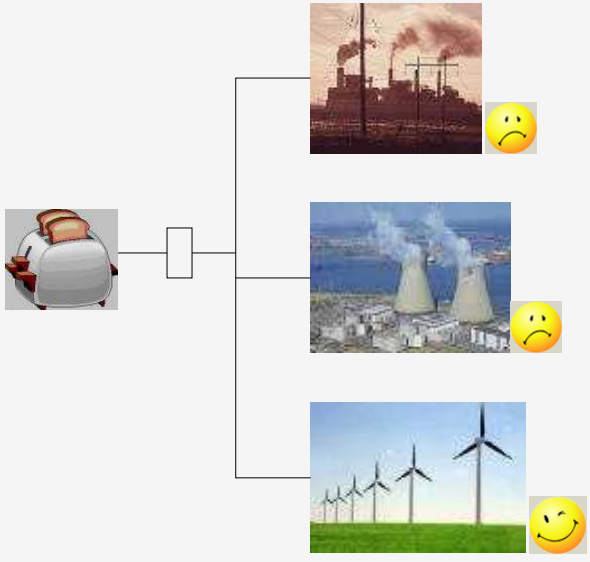
    ‘electrical power is interchangeable commodity’

(1) standard interface: electrical plug





grid computing

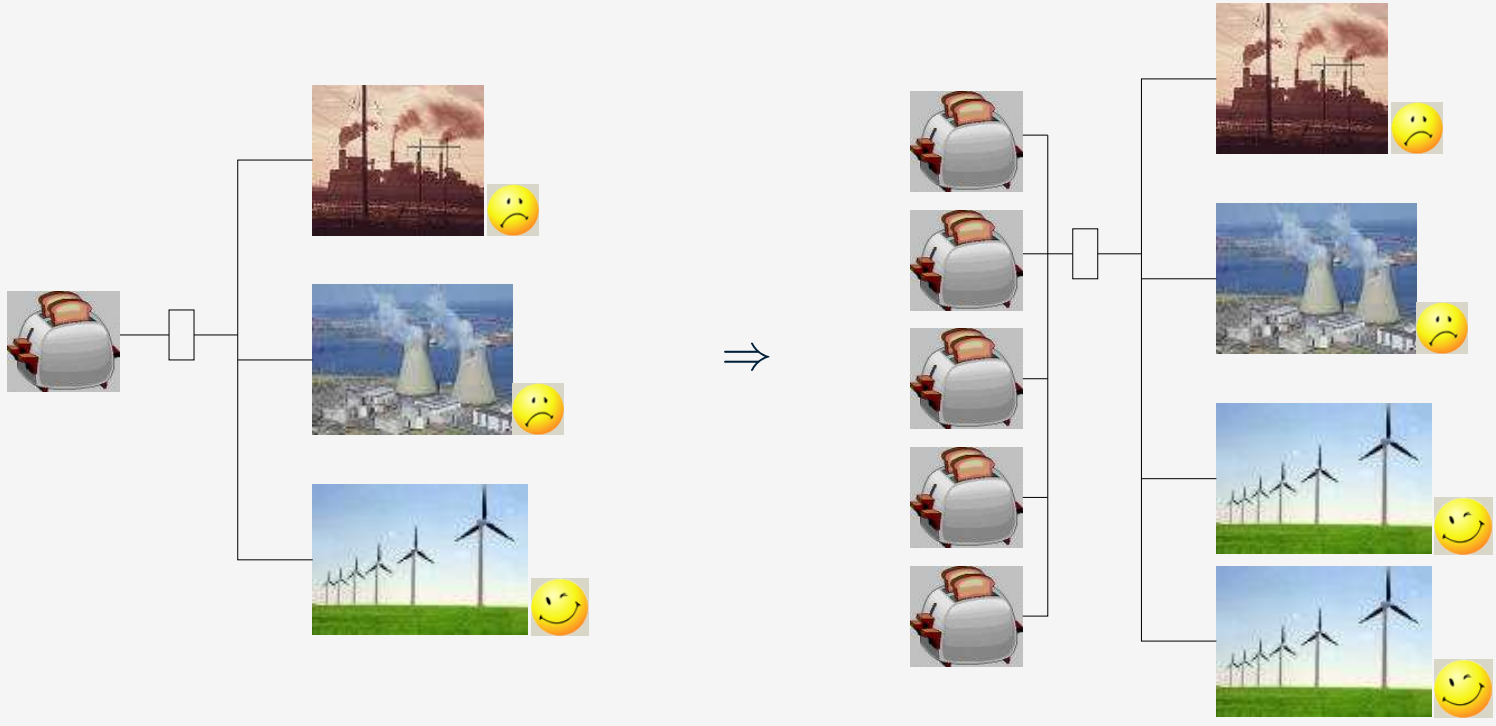


(2) scalable:

user: many appliances = many plugs

producer: when extra power is needed, switches in extra power plants

grid computing



(2) scalable:

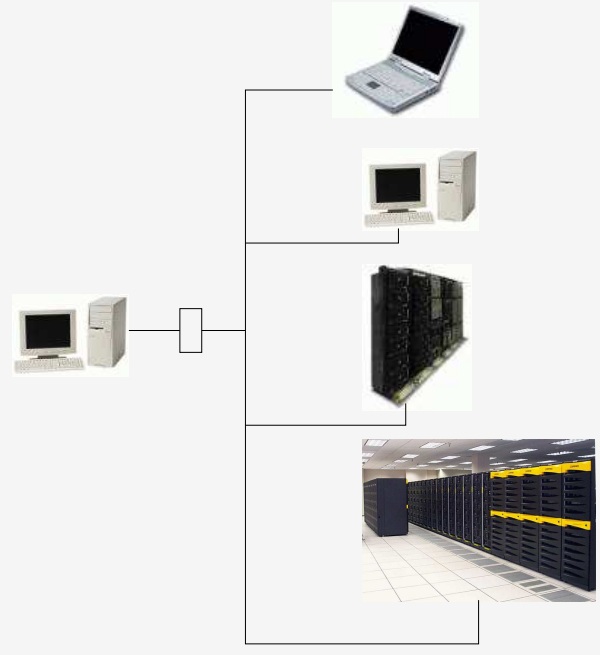
user: many appliances = many plugs

producer: when extra power is needed, switches in extra power plants

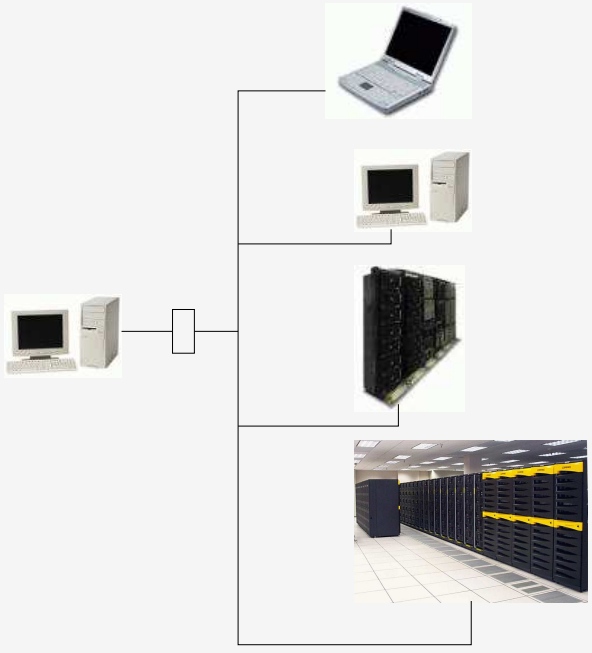
grid computing

- grid computing: analogous concept
  - user wants computing power
    - computing cycles
    - storage
    - network capacity
  - user doesn't care where the power is produced
    'computing power is inter-changeable commodity'

(1) standard interface



grid computing

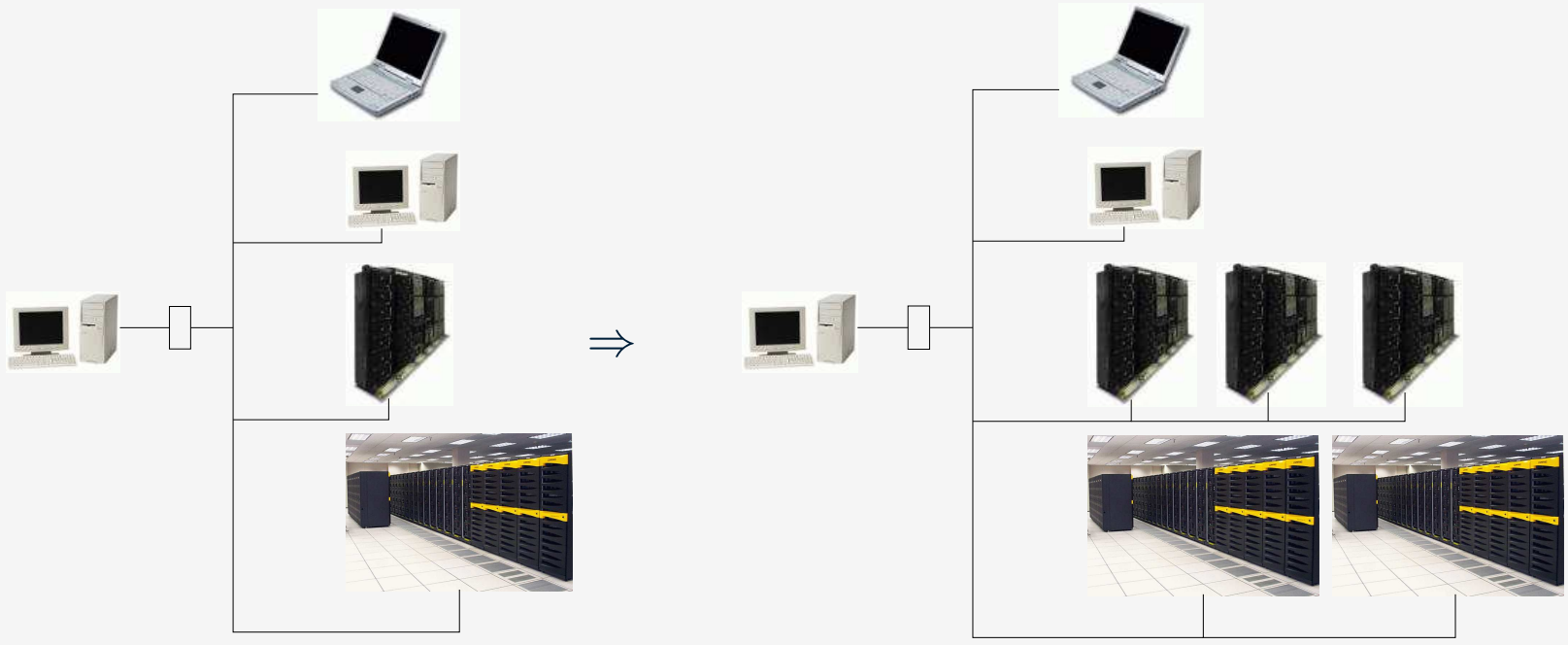


(2) scalable:

user: many applications at the same time

producer: when extra power is needed, switches in extra ‘computing plants’ (compute engines)

grid computing



(2) scalable:

user: many applications at the same time

producer: when extra power is needed, switches in extra ‘computing plants’ (compute engines)

grid computing

- the analogy is not perfect: computing power $\neq$ electrical power
- some additional requirements for computational grids:

(3) secure resource sharing
information is not anonymous (unlike electrical power)

(4) 'transactions', fault-tolerance
information is not replaceable (unlike electrical power)

(5) resource allocation, scheduling (queues)
large computers work with queueing systems

(6) install and compile application code and data on compute engines
variety of hardware, OS, OS and software versions, . . .

(7) communication between processors for parallel computing

(2) The TaskSpaces framework for grid computing

two major design choices:

(A) use the Java language

- concept of ‘virtual machine’:
  - behaves the same on all OS, hardware
  - ‘executable byte-code’ fully interchangeable
- has built-in security support
- object = data + code
- has built-in networking

$\Rightarrow$ provides natural and adequate answers to

(1) standard interface

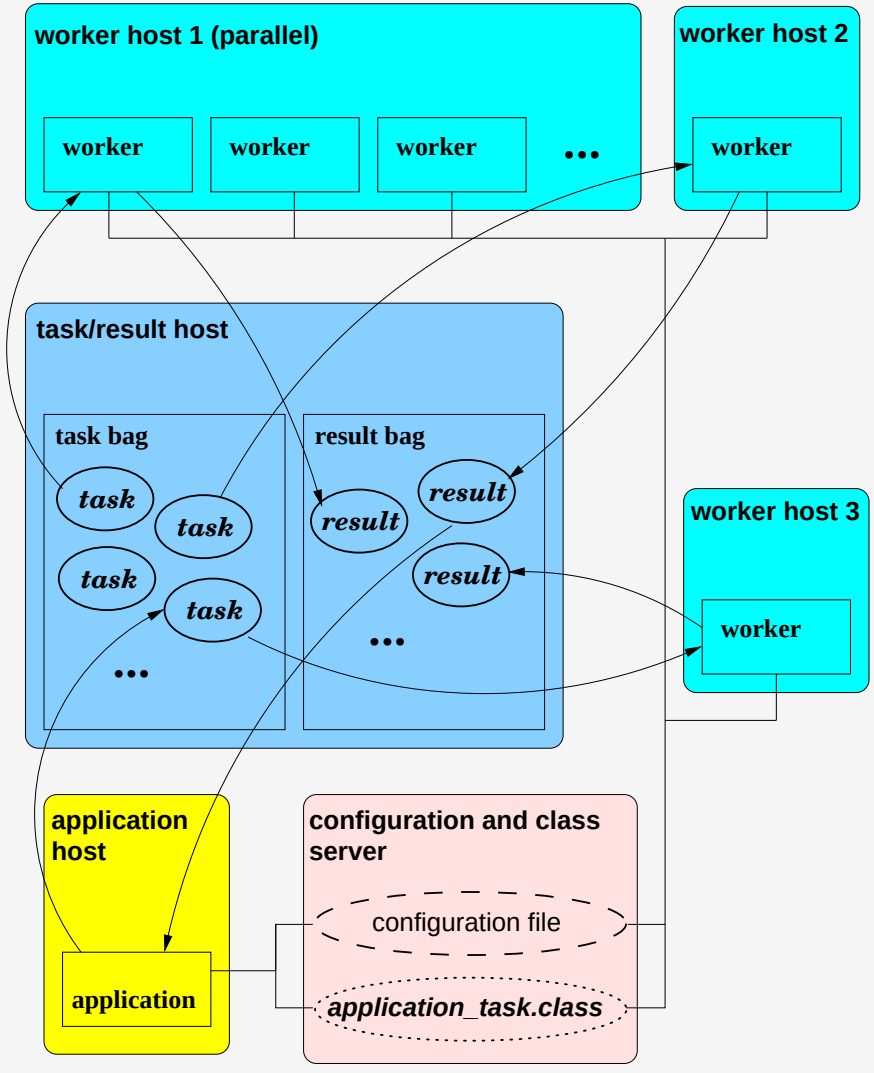
(3) secure resource sharing

(6) install and compile application code on compute engines: not needed! $\longrightarrow$ download task objects = code + data

(7) communication between processors: send objects

TaskSpaces: design choices

(B) use tuple space concept, bag-of-tasks paradigm



- space and time decoupling
- worker executable < 2 kb

$\Rightarrow$ provides natural and adequate answers to

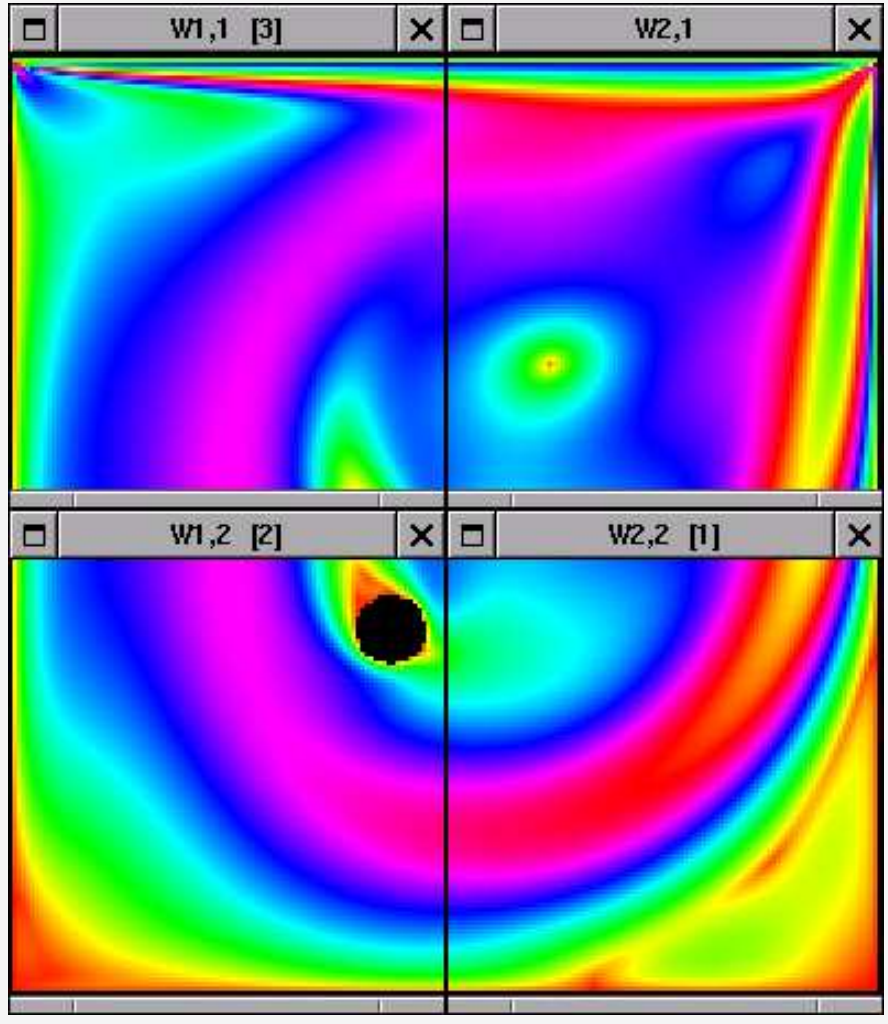
(2) scalable:

user: concurrent applications in one or several bags

producer: switches in extra 'worker farms'

(5) resource allocation, scheduling: task bag is simple decoupled 'superqueue'

application: interactive Lattice-Boltzmann simulation



< > - +

high throughput grid experiment

Blue Horizon, SDSC, San Diego, CA (4 workers/processor)	64	128	240
P4 Linux, CU Boulder, CO (2 workers/processor)	4	4	4
Itanium Linux, CU Boulder, CO (2 workers/processor)	4	4	4
forseti1, NCSA, Urbana, IL (1 worker/processor)	16	16	16
hermod, NCSA, Urbana, IL (1 worker/processor)	16	16	16
Total number of workers	104	168	280
Total execution time	105s	103s	101s

High throughput grid experiment (50 Jacobi iterations, 500x500 grid points per worker). Number of worker processes and total execution times are shown. Problem size is constant per worker process.

$\Longrightarrow$ TaskSpaces framework: parallel scientific computing with intertask communication is scalable on the internet!

TaskSpaces grid computing framework

(1) standard interface: Java task object

(2) scalable: through bag-of-tasks

user: many applications at the same time

producer: switches in extra ‘computing plants’

(3) secure resource sharing: digital certificates, to be implemented

(4) ‘transactions’, fault-tolerance: rudimentary, needs more thinking

(5) resource allocation, scheduling: task bag = decoupled superqueue

(6) get application code on compute engines: download task object

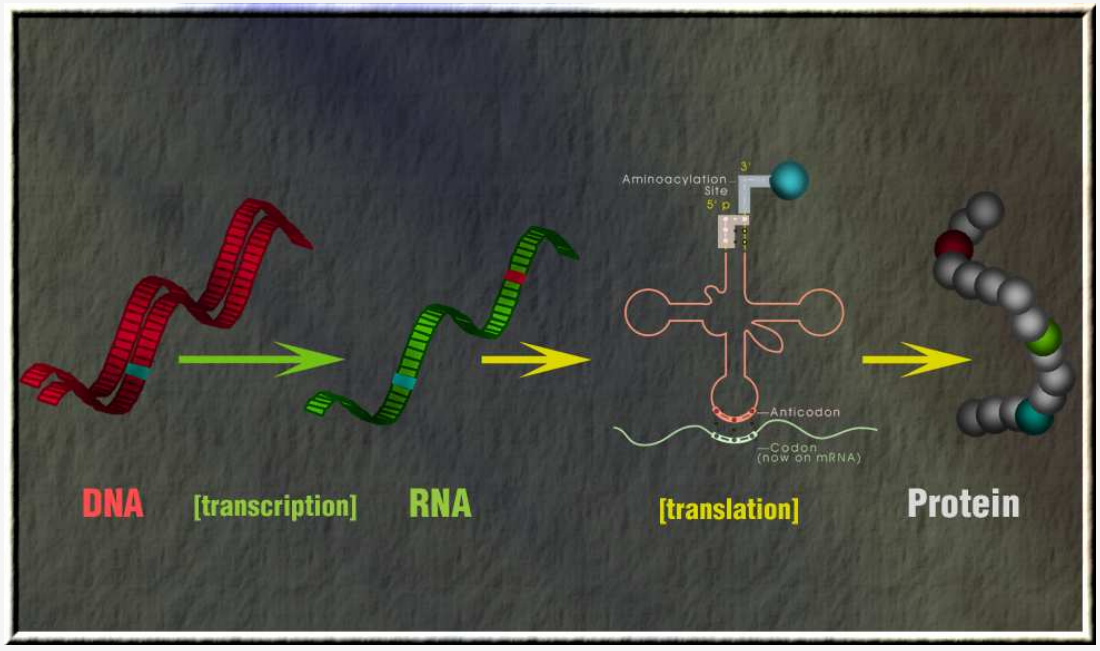
(7) interprocessor communication for parallel computing: send objects

(8) user gets charged according to use: to be implemented

(9) Quality of Service: by hand, to be automatized

(10) resource discovery: Globus?

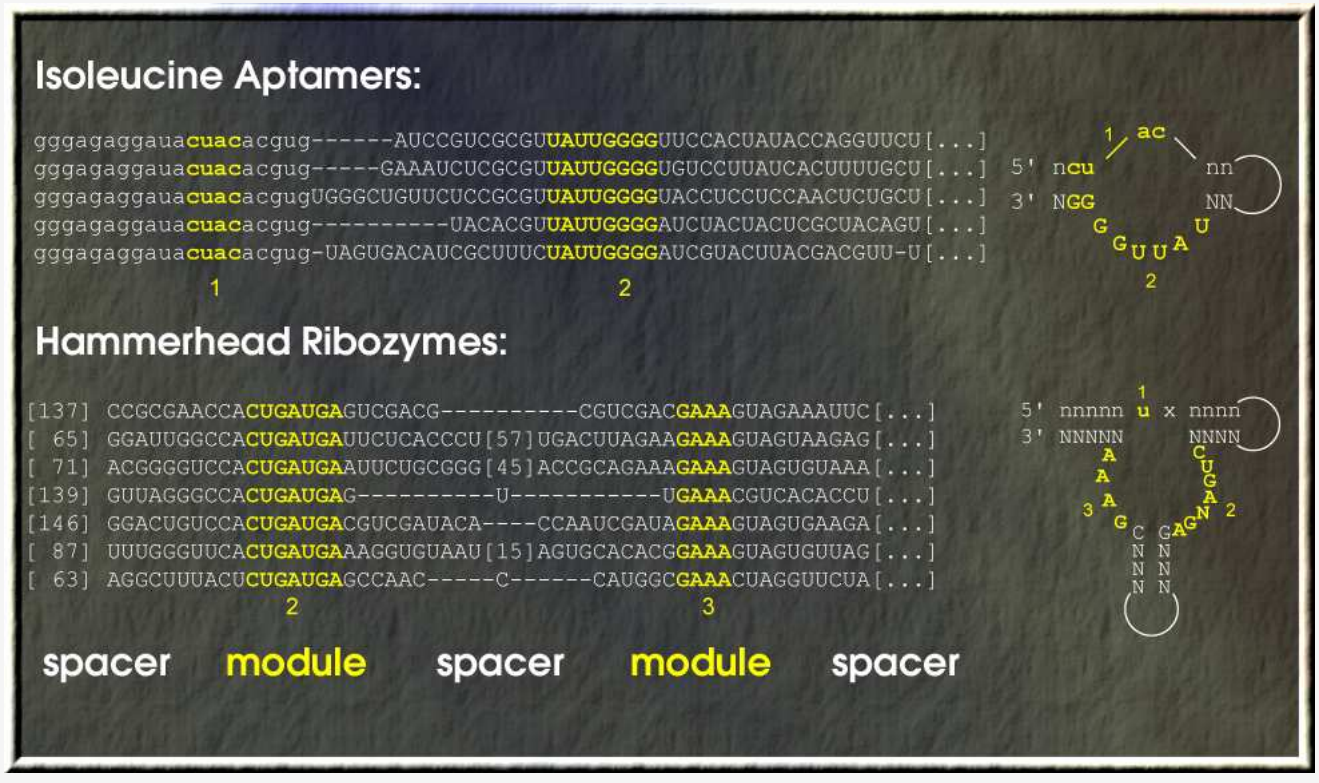
(3) Application: finding correctly folded active RNA motifs



- modern cells mostly use RNA molecules as *passive message*, but they also use RNA as an *active catalyst*
  - example: Hammerhead Ribozyme catalyzes cleavage of RNA
- RNA molecules that bind to specific molecules can experimentally be selected (SELEX)
  - example: Isoleucine Aptamer binds to Isoleucine aminoacid

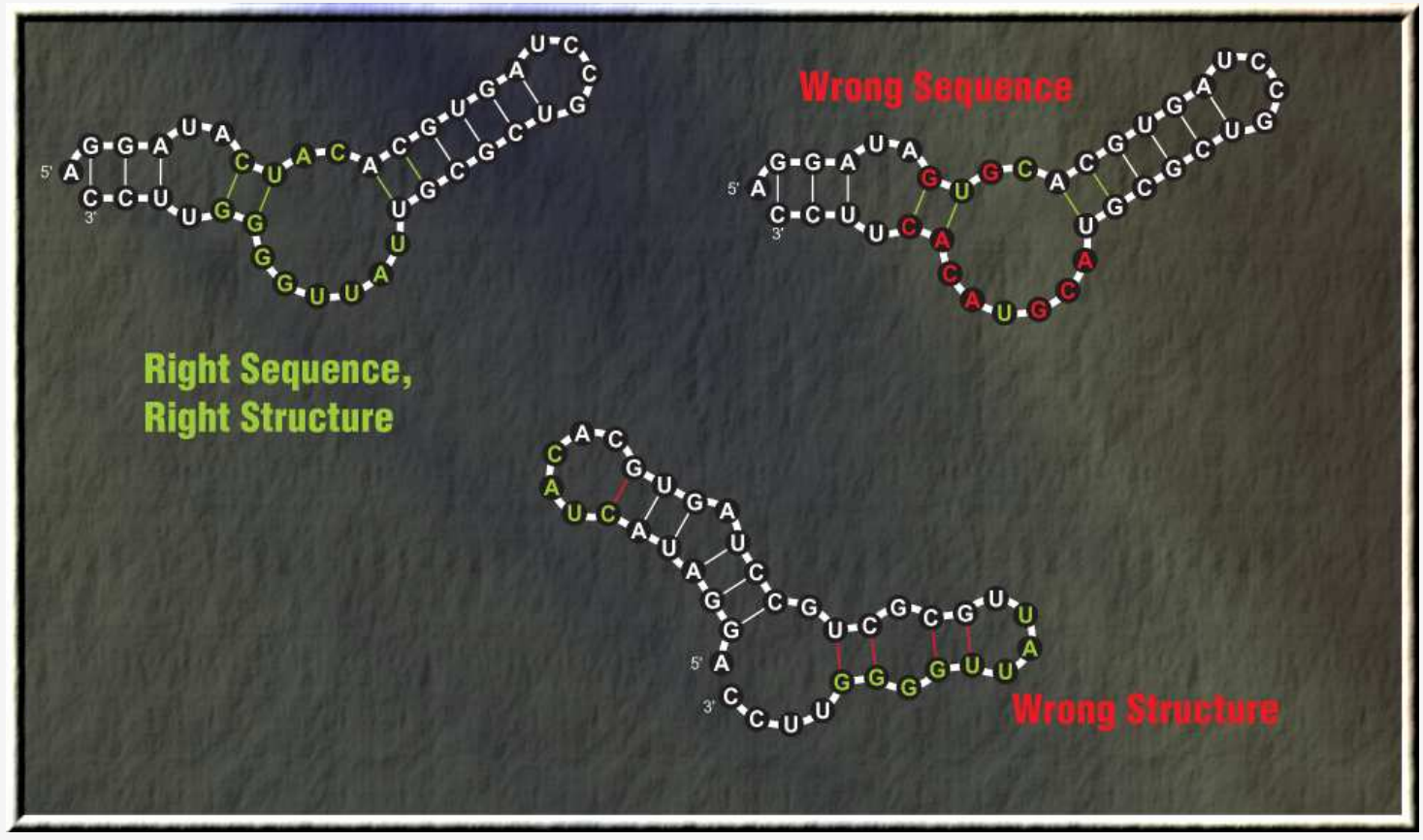
chemically actif motifs

- chemical activity characterized by specific *motif*
- *motif* has several *modules* with specified paired bases
- the *modules* are separated by arbitrary *spacer*



sequence and structure

- right sequence and right structure are both important



(Isoleucine aptamer)

finding correctly folded active RNA motifs

QUESTION:

given a random RNA molecule of a certain length (e.g. 50 nucleotides), what is the probability that this RNA molecule

- contains the (Isoleucine aptamer) motif in the sequence

and

- folds in the correct (Isoleucine aptamer) structure

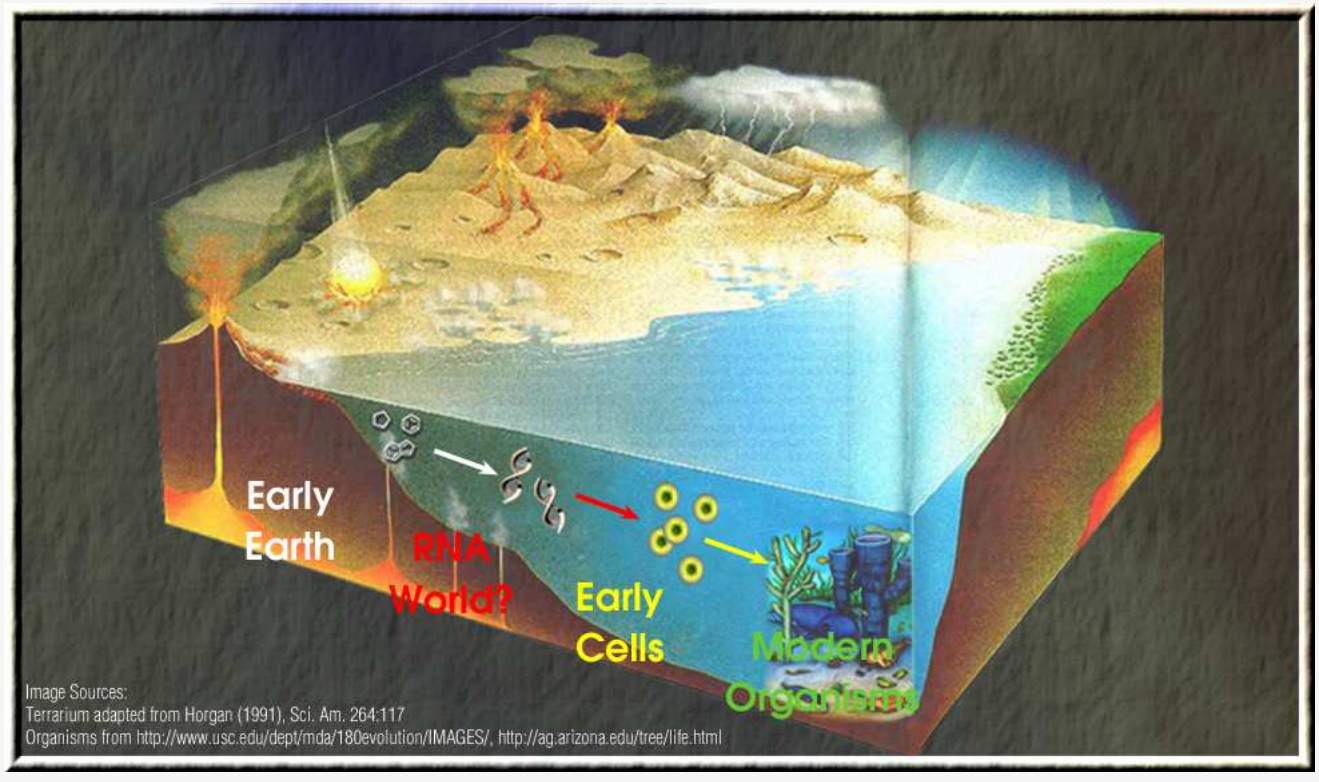
- applications:

  (1) by biasing the composition of the random molecule (e.g. more A, U than C, G), the probability to obtain an Isoleucine aptamer may be increased or decreased

  $\rightarrow$ possibly improve efficiency of SELEX by biasing composition of random pool (also: investigate effect of molecule length)

  (2) how likely is origin of an RNA world from a small number of random RNA molecules?

application: RNA world



if a small ($\sim 10^6$) number of random RNA molecules has a reasonable probability of containing molecules with various chemical functions, then a primitive metabolism may have originated from random RNA molecules

how to find this probability?

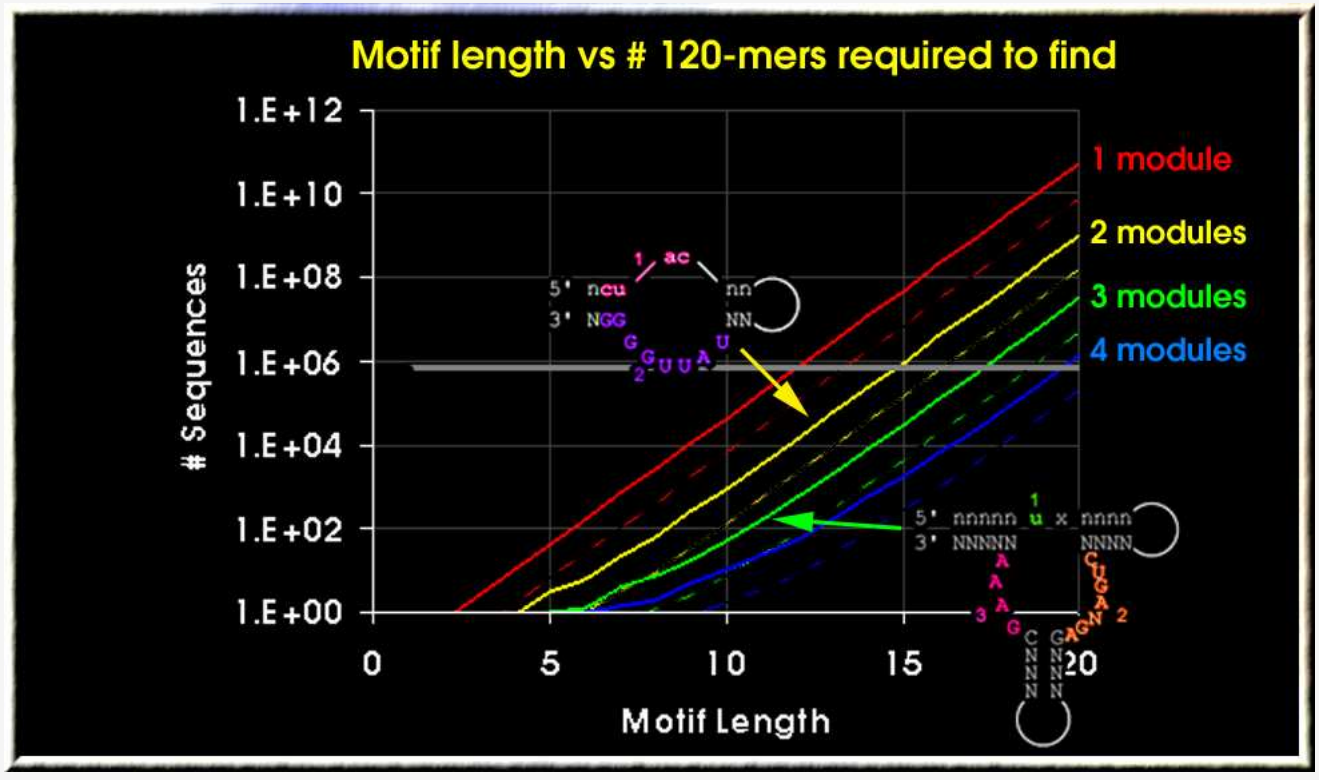
- $P(find\ correct\ sequence\ and\ structure) =$

  $P(sequence) \cdot P(structure|sequence)$

- Step 1: calculate $P(sequence)$

$P(sequence)$ can be approximated well by combinatorial formula that is easily computable

$P(sequence)$



- if only the completely specified nucleotides are taken into account, and not the structure, then only 10^5 random molecules are required to find the Isoleucine aptamer sequence, and only 10^2 to find the Hammerhead Ribozyme sequence!
- more modules increases probability of finding the motif sequence

Step 2: $P(structure|sequence)$

- $P(structure|sequence)$ cannot be estimated by closed formula because folding is a complex, nonlinear process
- estimate $P(structure|sequence)$ by simulation:
  fold RNA molecules that contain the correct motif sequence, but that are otherwise randomized, on a computer
- stepping in composition space, varying RNA length, and investigating many motifs, requires folding of billions of (short) RNA molecules

$\Rightarrow$ requires parallel bioinformatics

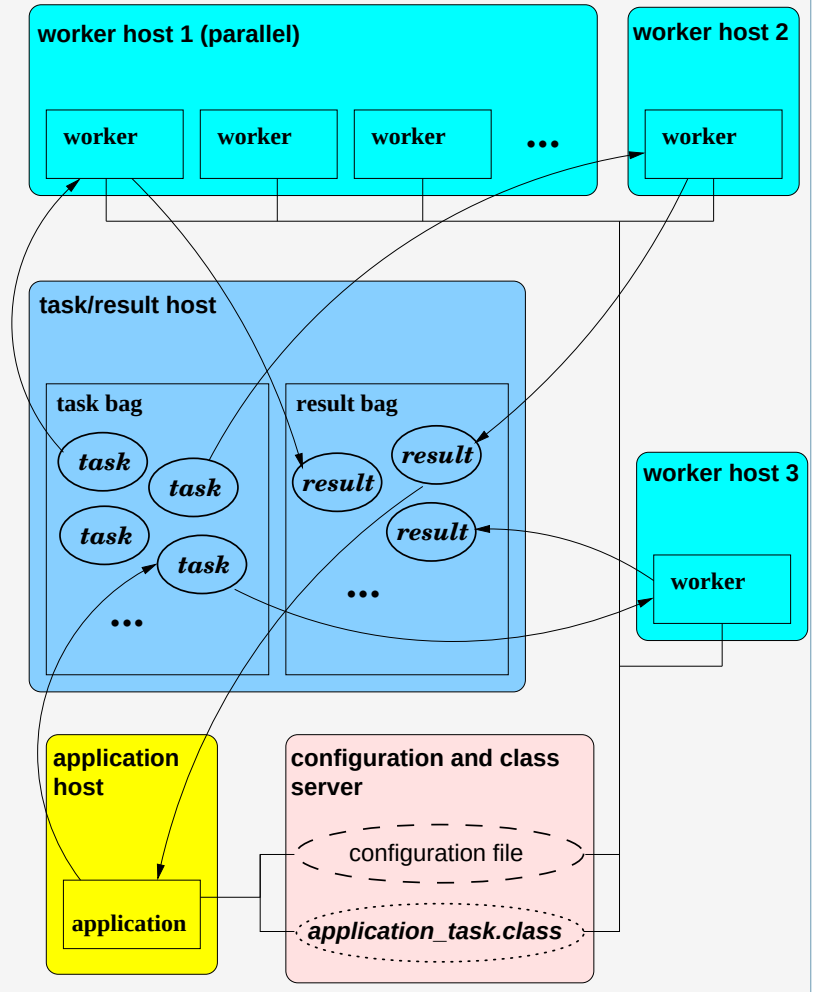
$\Rightarrow$ use TaksSpaces framework on computational grids

use TaskSpaces

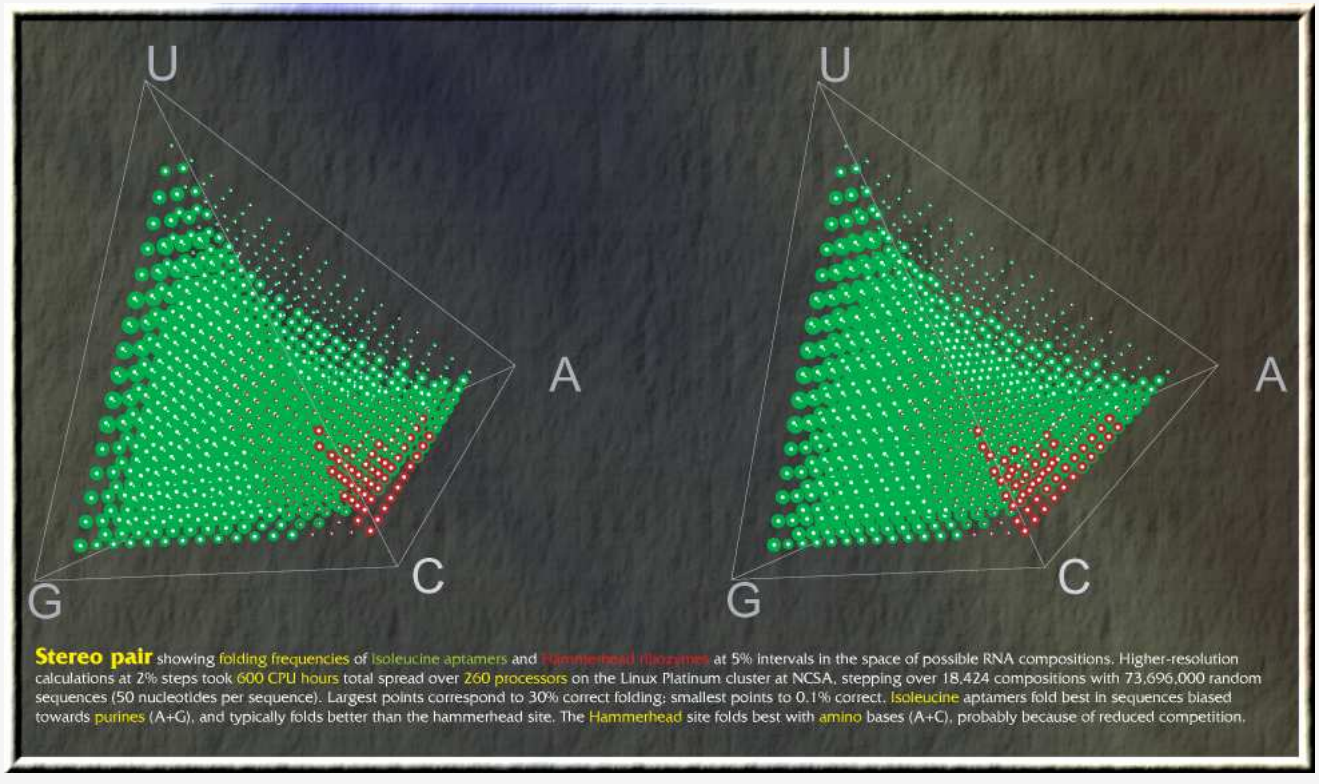
- example of one task = fold 10,000 randomized RNAs of length 50 that contain Isoleucine aptamer motif sequence, and count how many fold in the right structure
- use Vienna folding program
  - written in C
  - called from Java TaskSpaces framework on every processor
  - C needs to be compiled on every architecture
  - C executable can be downloaded with Java executable
- no communication between subtasks, so use TaskSpaces in taskfarming mode

TaskSpaces: example run

- 2% composition steps
  (=18,424 different compositions, tasks)
  50 nucleotides length
  2000 random sequences per composition
  $\rightarrow$ 73,696,000 foldings
- use Linux Cluster Platinum at NCSA (Illinois)
  $\sim$ 1000 processors
- use 260 processors, not all concurrently
  600 CPU hours
  (25 days on one CPU)

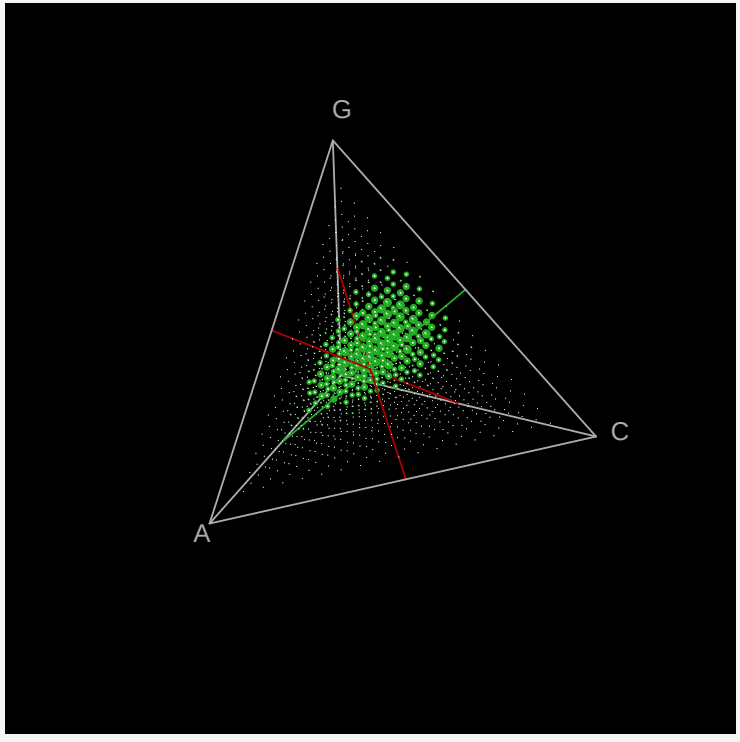


results: $P(structure|sequence,\ composition)$



- Isoleucine aptamer folds best when sequence composition is biased toward purines (A and G)
- Hammerhead ribozyme folds best when sequence composition is biased toward amino bases (A and C)

$$P(sequence\ and\ structure) = P(sequence) \cdot P(structure|sequence)$$



preliminary results, length 50

- Ile best at (15%, 30%, 35%, 20%) ACGU composition, factor 1.56 better than uniform composition
- HH best at (20%, 30%, 40%, 10%) ACGU composition, factor 2.27 better than uniform composition

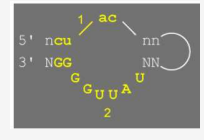
- RNA world: $10^9 - 10^{10}$ random 50-mers needed for Ile
- RNA world: $10^{12} - 10^{13}$ random 50-mers needed for HH

$\Rightarrow$ we have to study longer molecules (longer computing times!), and allow for GU pairs; probabilities will be larger

(4) Conclusions and future work

- conclusions:
  - TaskSpaces framework has been developed for parallel scientific computing on computational grids
  - TaskSpaces has been applied to parallel bioinformatics problem: finding correctly folded active RNA motifs

  preliminary conclusions for RNA application:
  - SELEX may be optimized somewhat by biasing composition, but the results show that equal base frequencies are within 2-fold of the optimum
  - origin of RNA world from relatively small number of random RNA molecules may gain support from our calculations

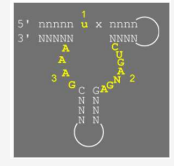


Conclusions and future work

- work to be done:
  - continue development of TaskSpaces (modularity, security, fault-tolerance, . . .)
  - RNA application:
    - need to study longer RNA molecules
    - need to study more motifs

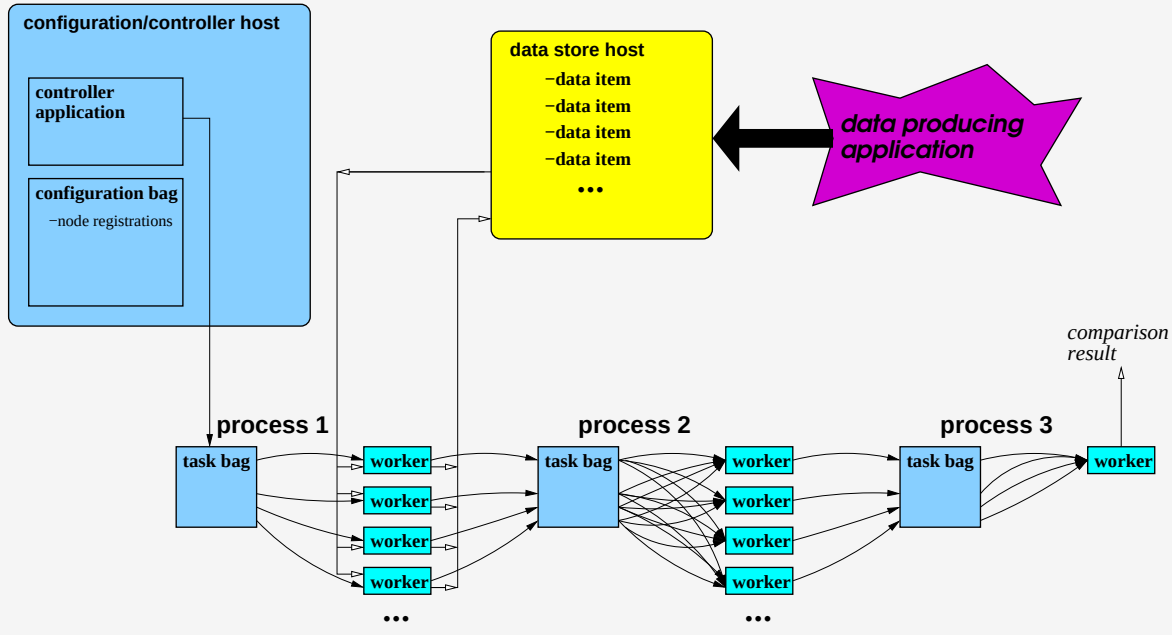
      database is being developed with $\sim 50 - 100$ motifs, in collaboration with Fang En Lee (U Texas) and Erik Schultes (Whitehead Institute)

      $\rightarrow$ need to fold billions of RNA molecules; need lots of computing power!
    - SELEX experimental verification in progress or planned



Conclusions and future work

- future work:
  - other parallel bioinformatics applications for TaskSpaces
  - extend TaskSpaces
    - parallel workflows for data processing
    - possible application: proteomics processing



A software framework for parallel bioinformatics on computational grids

Hans De Sterck[1], Rob Markel[2,3,4], and Rob Knight[3]

[1]Department of Applied Mathematics, CU Boulder (desterck@colorado.edu)

[2]Department of Interdisciplinary Telecommunications, CU Boulder

[3]Yarus Lab, MCDB, CU Boulder

[4]Scientific Computing Division, NCAR

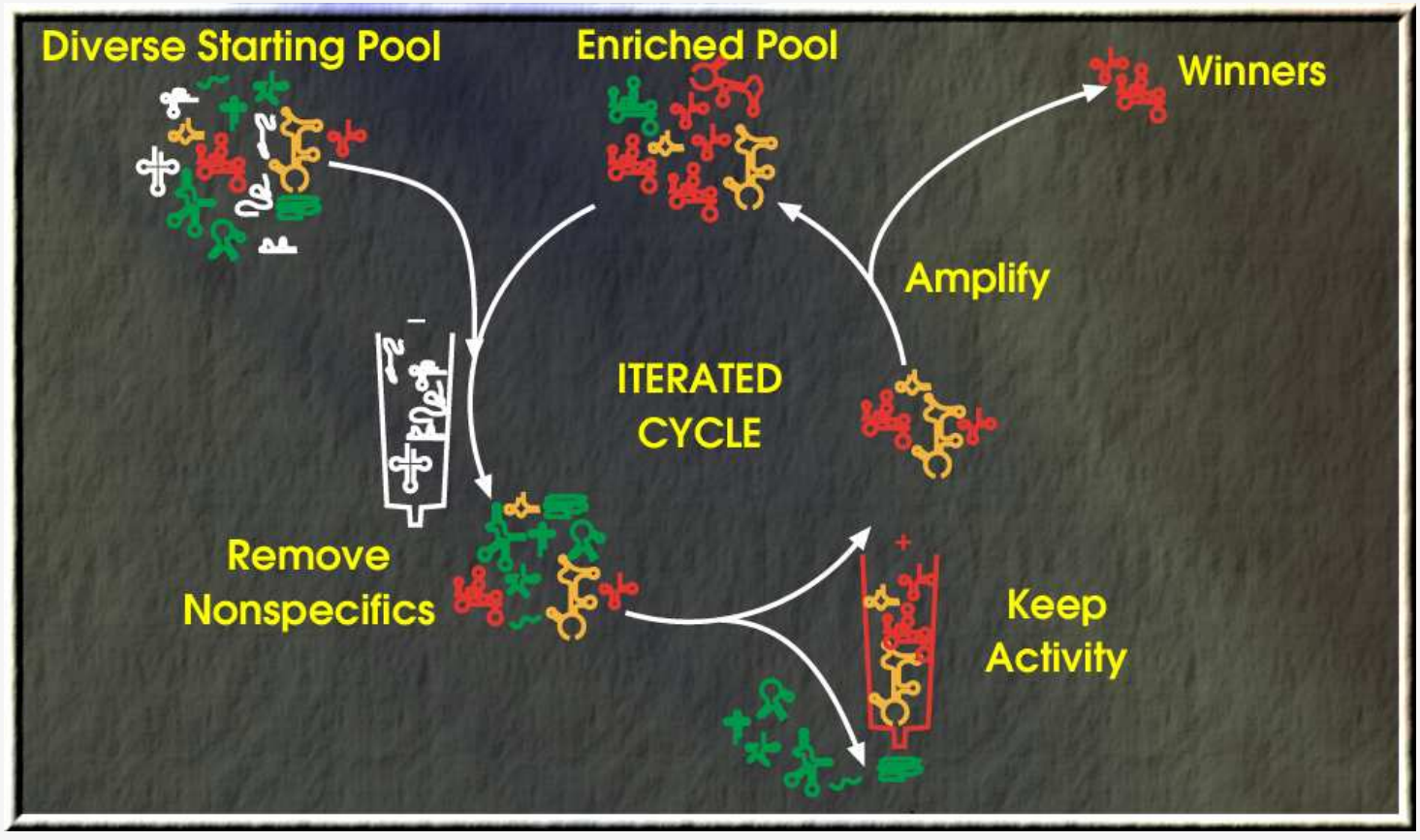
Bioinformatics Supergroup

Monday, November 3, 2003

NSF-funded computing resources

- SDSC, San Diego:
  www.npaci.edu/Allocations
- NCSA, Illinois:
  www.ncsa.uiuc.edu/UserInfo/Allocations/Overview.html
- PSC, Pittsburgh:
  www.psc.edu/work_with_us.html

SELEX


Diverse Starting Pool
Enriched Pool
Winners
Amplify
ITERATED
CYCLE
Remove
Nonspecifics
Keep
Activity

grid computing: efficiency

- some tasks run more efficiently on computer A than on computer B,
  especially when ‘tuned’ for computer A
  $\rightarrow$ efficiency price is paid for interchangeability
  but: keep important advantages of scalability, interchangeability
- still, it may be useful to choose somewhat ‘specialized’ resources

(10) resource discovery

grid computing

(1) standard interface

(2) scalable:

user: many applications at the same time

producer: switches in extra ‘computing plants’

(3) user gets charged according to use

(4) Quality of Service

(5) secure resource sharing

(6) ‘transactions’, fault-tolerance

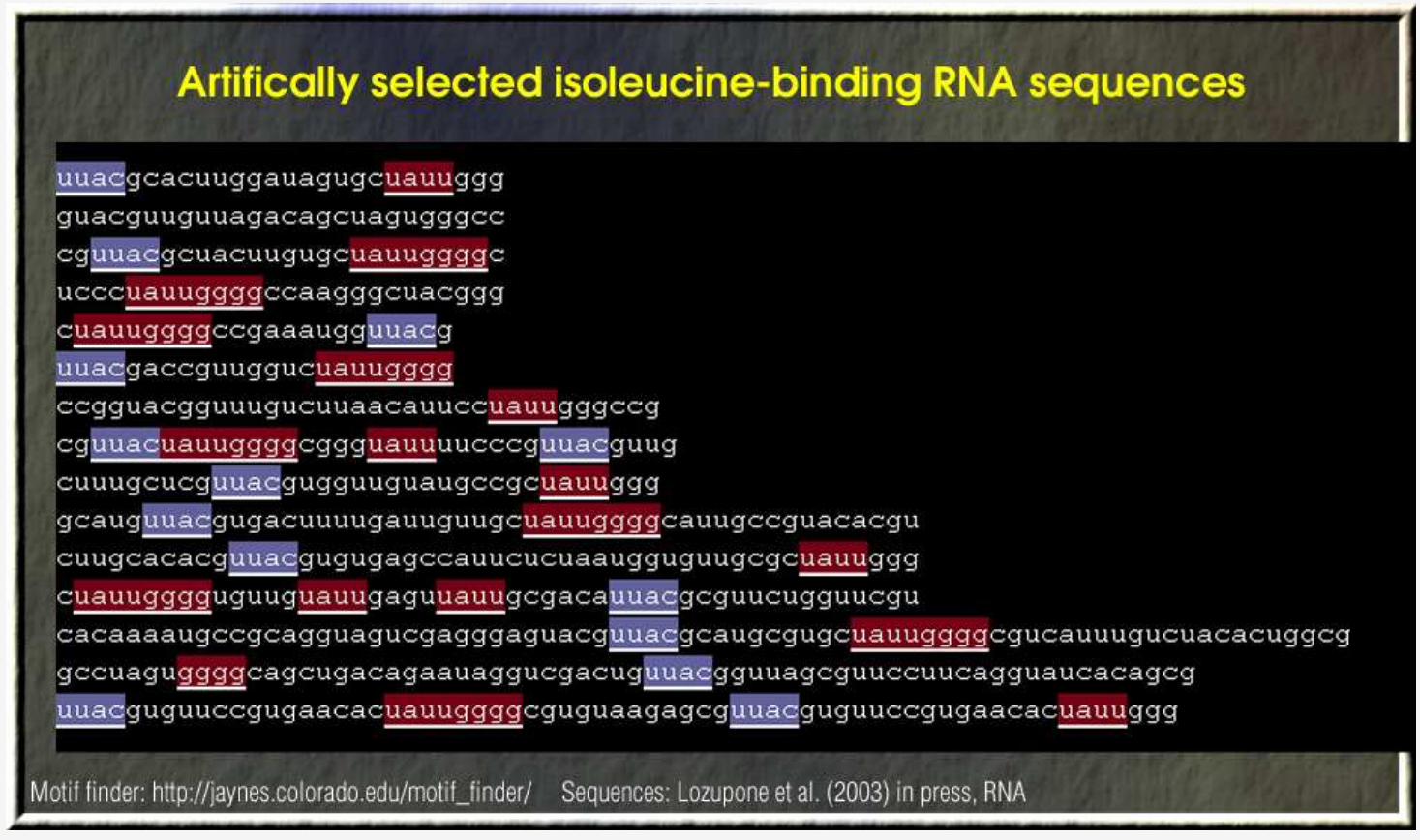
(7) resource allocation, scheduling (queues)

(8) install and compile application code on compute engines

(9) communication between processors for parallel computing

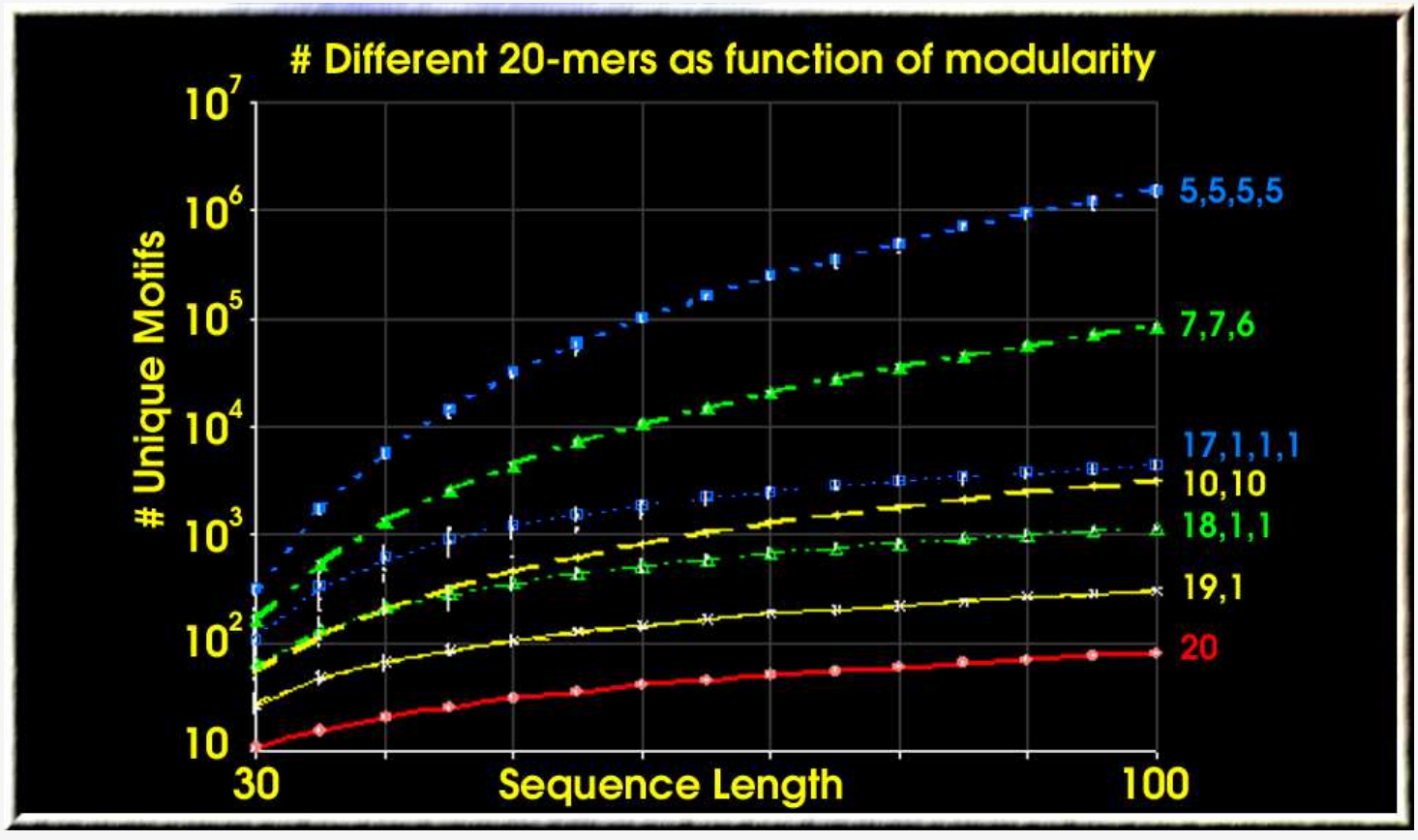
(10) resource discovery

SELEX experiment



- chemically actif *motif* with several *modules*

$P(sequence)$



- More modules evenly divided can be easier to find than fewer modules unevenly divided. Predictions are quantitative.