SparkulAkka: a Simple and Extensible Distributed Simulation System built on Spark and Akka

WSU Vancouver

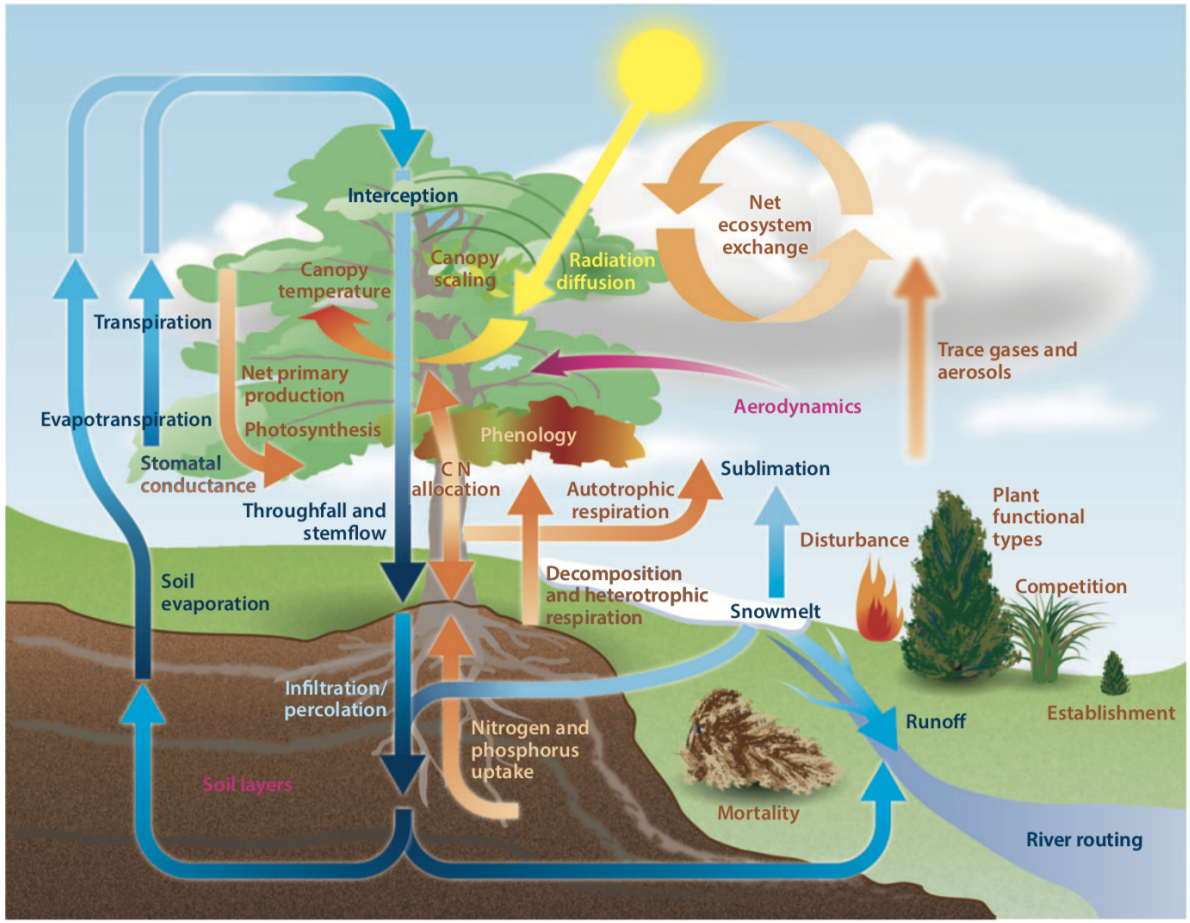
Fabiana Ferracina

April 23, 2018

Overview

My research focus is on the modeling of forest dynamics and processes.

My research focus is on the modeling of forest dynamics and processes. My motivation is to use mathematics and computer science tools to model everything here:

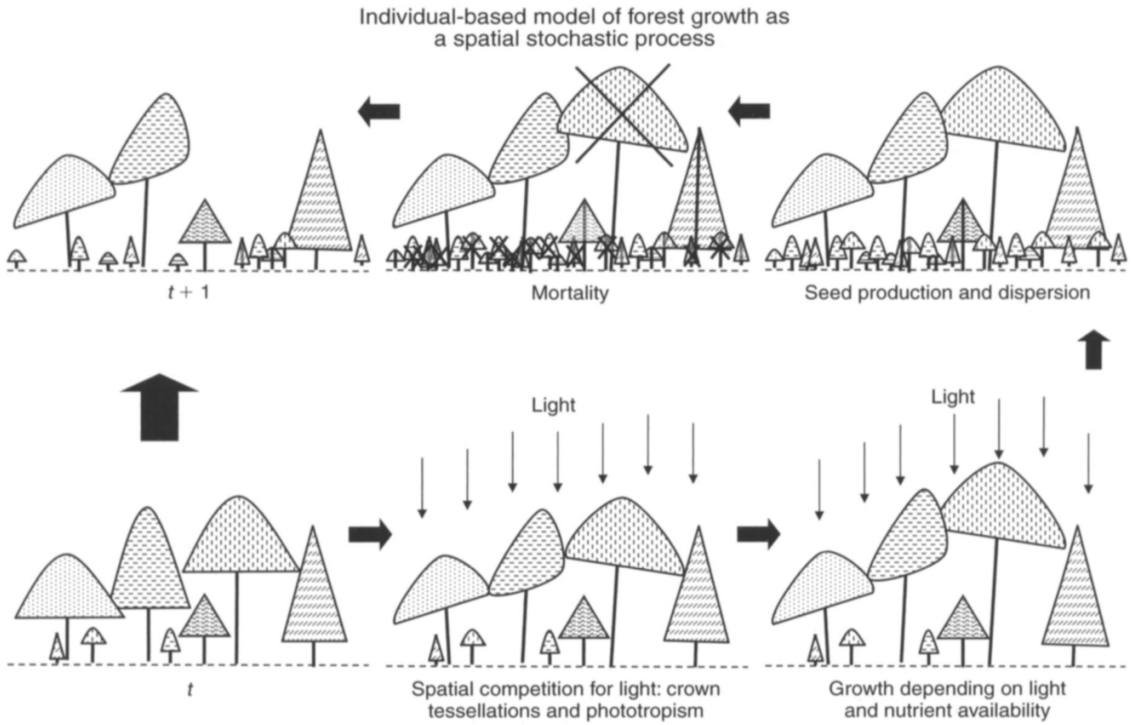


One step at a time...

Let's focus on succession dynamics for now:

One step at a time...

Let's focus on succession dynamics for now:



Many Questions!

- How do we implement sustainable forestry?

Many Questions!

- How do we implement sustainable forestry?
- How long will it take for a forest to recover after a large fire?

Many Questions!

- How do we implement sustainable forestry?
- How long will it take for a forest to recover after a large fire?
- What will happen to the Northwest forests if temperatures increase by 2 degrees Celsius?

Many Questions!

- How do we implement sustainable forestry?
- How long will it take for a forest to recover after a large fire?
- What will happen to the Northwest forests if temperatures increase by 2 degrees Celsius?

Data can help, also a computer or 50...

Overview

Simulations are computed scenarios based on a simplified version of reality usually captured by differential equations.

Simulations are computed scenarios based on a simplified version of reality usually captured by differential equations.

Natural processes are usually stochastic and non-linear.

Simulations are computed scenarios based on a simplified version of reality usually captured by differential equations.

Natural processes are usually stochastic and non-linear.

We want to compute new values, evolve the system given these values, and loop for thousands of cycles over a large area containing millions of entities

Simulations are computed scenarios based on a simplified version of reality usually captured by differential equations.

Natural processes are usually stochastic and non-linear.

We want to compute new values, evolve the system given these values, and loop for thousands of cycles over a large area containing millions of entities

... and we want to use previous data and keep history...

Simulations are computed scenarios based on a simplified version of reality usually captured by differential equations.

Natural processes are usually stochastic and non-linear.

We want to compute new values, evolve the system given these values, and loop for thousands of cycles over a large area containing millions of entities

... and we want to use previous data and keep history...

Interestingly, active forest modeling research, despite the many mentions of computational complexity and data-intensiveness has a gap when it comes to distributed computing.

Overview

- An approach to understand and reason about concurrency.

- An approach to understand and reason about concurrency.
- Executing objects, called Actors, communicate exclusively through asynchronous messages.

- An approach to understand and reason about concurrency.
- Executing objects, called Actors, communicate exclusively through asynchronous messages.
- State is maintained privately within each Actor, thus deadlocks and data integrity pose little to no issues in an Actor-oriented system.

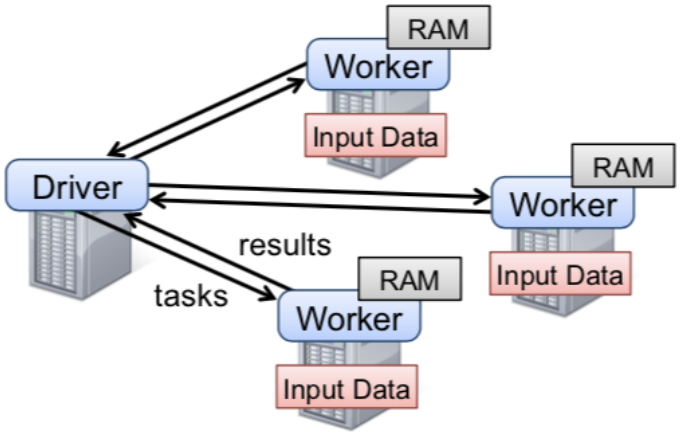
Overview

- Spark - an in-memory distributed data analysis framework.

- Spark - an in-memory distributed data analysis framework.
- RDDs: Resilient Distributed Datasets - a distributed memory abstraction.

- Spark - an in-memory distributed data analysis framework.
- RDDs: Resilient Distributed Datasets - a distributed memory abstraction.
- RDDs allow for fault-tolerant computations on large clusters by providing a restricted form of shared memory based on coarse-grained transformations.

- Spark - an in-memory distributed data analysis framework.
- RDDs: Resilient Distributed Datasets - a distributed memory abstraction.
- RDDs allow for fault-tolerant computations on large clusters by providing a restricted form of shared memory based on coarse-grained transformations.
- Spark Runtime:



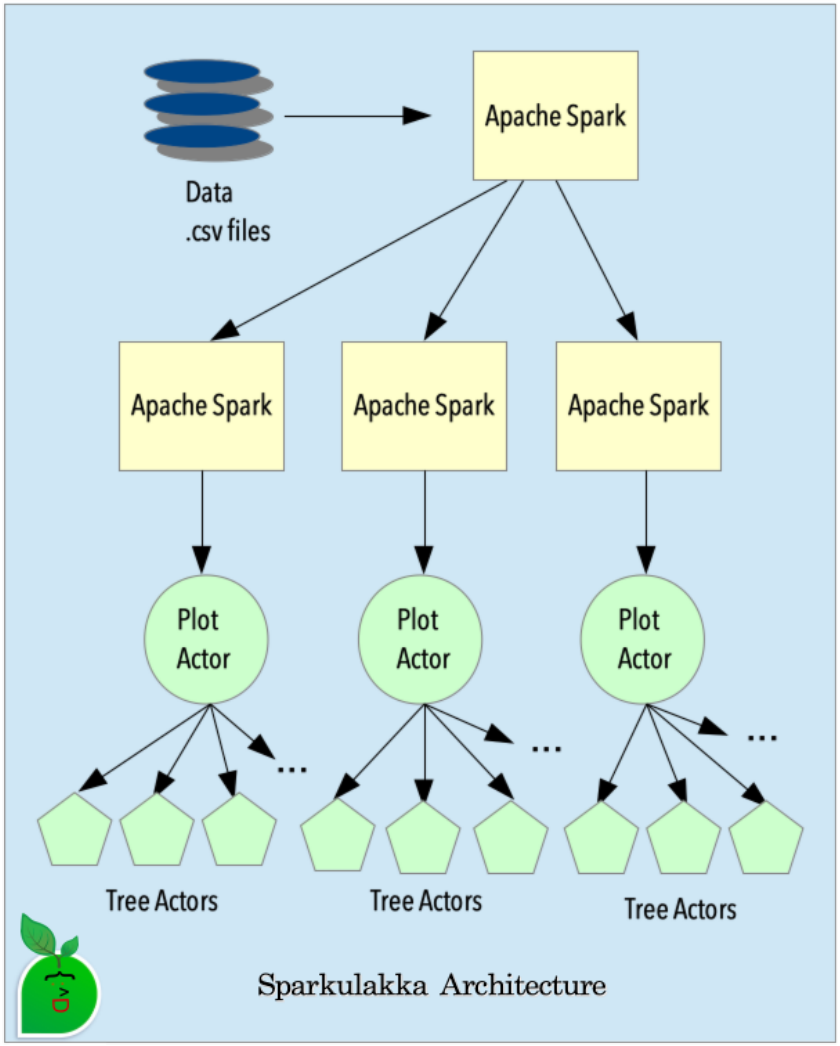


Figure: SparkulAkka architecture and a Harvard Forest flux tower

Overview

- Spark to load, process and analyze forest measurement data - Sparkulation piece where further statistical models can be adapted.

- Spark to load, process and analyze forest measurement data - Sparkulation piece where further statistical models can be adapted.
- Data sources: FIADB, WorldClim, Harvard Forest flux towers

- Spark to load, process and analyze forest measurement data - Sparkulation piece where further statistical models can be adapted.
- Data sources: FIADB, WorldClim, Harvard Forest flux towers
- Sample Spark SQL:

```
SELECT plot, spcd, AVG(ht), AVG(dia)
FROM IL_TREE_csv
WHERE statuscd = 1 AND diahtcd = 1 AND ht IS NOT NULL
    AND invyr = "2008"
GROUP BY 1, 2 ORDER BY 1, 2;
```

- Spark to load, process and analyze forest measurement data - Sparkulation piece where further statistical models can be adapted.
- Data sources: FIADB, WorldClim, Harvard Forest flux towers
- Sample Spark SQL:

```
SELECT plot, spcd, AVG(ht), AVG(dia)
FROM IL_TREE_csv
WHERE statuscd = 1 AND diahtcd = 1 AND ht IS NOT NULL
    AND invyr = "2008"
GROUP BY 1, 2 ORDER BY 1, 2;
```

- WHERE conditions are filter operations. GROUP BY is a groupByKey (with reduce for calculating the final values via AVG). ORDER BY is a sort.

- Initial differential model variables are then fed into a simulation using the Actor Model framework Akka.

- Initial differential model variables are then fed into a simulation using the Actor Model framework Akka.
- As we can see Actors fully utilize available cores:

```
Processes: 295 total, 3 running, 292 sleeping, 1216 threads                                                        20:55:53
Load Avg: 4.79, 2.26, 1.89  CPU usage: 98.28% user, 1.71% sys, 0.0% idle     SharedLibs: 213M resident, 63M data, 26M linkedit.
MemRegions: 26043 total, 2365M resident, 165M private, 523M shared. PhysMem: 7747M used (1466M wired), 444M unused.
VM: 1321G vsize, 1297M framework vsize, 0(0) swapins, 0(0) swapouts. Networks: packets: 84178/42M in, 67523/12M out.
Disks: 245300/6196M read, 133444/4176M written.

PID  COMMAND      %CPU  TIME     #TH   #WQ  #PORT MEM    PURG  CMPRS  PGRP PPID STATE    BOOSTS          %CPU_ME %CPU_OTHRS UID  FAULTS
922  screencaptur 0.1   00:00.16 4     2    152   3712K  0B    0B     922  1    sleeping *0[1634+]       0.00000 0.04138    501  5498
921  screencaptur 1.2   00:01.13 2     1    54    3304K  636K  0B     308  308  sleeping *0[1]           0.04138 0.00000    501  112271+
920  java         392.1 01:03.51 41/8  1    127   682M+  0B    0B     453  453  running  *0[1]           0.00000 0.00000    501  187405+
919  java         0.0   00:03.06 29    1    103   144M   0B    0B     453  453  sleeping *0[1]           0.00000 0.00000    501  42976
918  java         0.0   00:01.45 33    1    111   74M    0B    0B     453  1    sleeping *0[1]           0.00000 0.00000    501  24572
917  ocspd        0.0   00:00.02 2     1    27    1340K  0B    0B     917  1    sleeping *0[1]           0.00000 0.00000    0    2131
916  java         0.0   00:09.66 31    1    107   198M   0B    0B     453  453  sleeping *0[1]           0.00000 0.00000    501  91106
911  mdworker_sha 0.0   00:00.04 3     1    59    3072K  0B    0B     911  1    sleeping *0[1]           0.00000 0.00000    501  3778
910  mdworker_sha 0.0   00:00.04 3     1    59    3112K  0B    0B     910  1    sleeping *0[1]           0.00000 0.00000    501  3784
```

- Initial differential model variables are then fed into a simulation using the Actor Model framework Akka.
- As we can see Actors fully utilize available cores:

Processes: 295 total, 3 running, 292 sleeping, 1216 threads 20:55:53
Load Avg: 4.79, 2.26, 1.89 CPU usage: 98.28% user, 1.71% sys, 0.0% idle SharedLibs: 213M resident, 63M data, 26M linkedit.
MemRegions: 26043 total, 2365M resident, 165M private, 523M shared. PhysMem: 7747M used (1466M wired), 444M unused.
VM: 1321G vsize, 1297M framework vsize, 0(0) swapins, 0(0) swapouts. Networks: packets: 84178/42M in, 67523/12M out.
Disks: 245300/6196M read, 133444/4176M written.

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS	UID	FAULTS
922	screencaptur	0.1	00:00.16	4	2	152	3712K	0B	0B	922	1	sleeping	*0[1634+]	0.00000	0.04138	501	5498
921	screencaptur	1.2	00:01.13	2	1	54	3304K	636K	0B	308	308	sleeping	*0[1]	0.04138	0.00000	501	112271+
920	java	392.1	01:03.51	41/8	1	127	682M+	0B	0B	453	453	running	*0[1]	0.00000	0.00000	501	187405+
919	java	0.0	00:03.06	29	1	103	144M	0B	0B	453	453	sleeping	*0[1]	0.00000	0.00000	501	42976
918	java	0.0	00:01.45	33	1	111	74M	0B	0B	453	1	sleeping	*0[1]	0.00000	0.00000	501	24572
917	ocspd	0.0	00:00.02	2	1	27	1340K	0B	0B	917	1	sleeping	*0[1]	0.00000	0.00000	0	2131
916	java	0.0	00:09.66	31	1	107	198M	0B	0B	453	453	sleeping	*0[1]	0.00000	0.00000	501	91106
911	mdworker_sha	0.0	00:00.04	3	1	59	3072K	0B	0B	911	1	sleeping	*0[1]	0.00000	0.00000	501	3778
910	mdworker_sha	0.0	00:00.04	3	1	59	3112K	0B	0B	910	1	sleeping	*0[1]	0.00000	0.00000	501	3784

- Whereas the sequential implementation does not:

Processes: 279 total, 3 running, 276 sleeping, 1098 threads 19:25:59
Load Avg: 1.56, 1.58, 1.71 CPU usage: 25.80% user, 2.7% sys, 72.11% idle SharedLibs: 288M resident, 63M data, 37M linkedit.
MemRegions: 25359 total, 2449M resident, 158M private, 446M shared. PhysMem: 7362M used (1424M wired), 829M unused.
VM: 1254G vsize, 1297M framework vsize, 0(0) swapins, 0(0) swapouts. Networks: packets: 22615/6950K in, 19320/4691K out.
Disks: 193532/4868M read, 91764/3399M written.

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPR	PGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS	UID	FAULTS
666	screencaptur	0.0	00:00.31	3	1	148	3576K	0B	0B	666	1	sleeping	*0[3113]	0.00000	0.00000	501	5515
665	screencaptur	0.0	00:02.71	2	1	56	3360K	636K	0B	308	308	sleeping	*0[1]	0.00000	0.00000	501	276900
664	Python	99.5	00:41.89	1/1	0	10	15M	0B	0B	664	632	running	*0[1]	0.00000	0.00000	501	5093
650	com.apple.iC	0.0	00:00.05	2	1	55	3072K	0B	0B	650	1	sleeping	0[1]	0.00000	0.00000	501	4396
632	bash	0.0	00:00.04	1	0	21	892K	0B	0B	632	631	sleeping	*0[1]	0.00000	0.00000	501	1524
631	login	0.0	00:00.01	2	1	31	1032K	0B	0B	631	488	sleeping	*0[9]	0.00000	0.00000	0	1741
630	mdworker_sha	0.0	00:00.07	4	1	61	4580K	0B	0B	630	1	sleeping	*0[1]	0.00000	0.00000	501	8055
624	trustd	0.0	00:00.09	2	2	37	4188K	168K	0B	624	1	sleeping	0[35]	0.00000	0.00000	89	3012

- Initial differential model variables are then fed into a simulation using the Actor Model framework Akka.
- As we can see Actors fully utilize available cores:

```
Processes: 295 total, 3 running, 292 sleeping, 1216 threads                                                            20:55:53
Load Avg: 4.79, 2.26, 1.89  CPU usage: 98.28% user, 1.71% sys, 0.0% idle     SharedLibs: 213M resident, 63M data, 26M linkedit.
MemRegions: 26043 total, 2365M resident, 165M private, 523M shared. PhysMem: 7747M used (1466M wired), 444M unused.
VM: 1321G vsize, 1297M framework vsize, 0(0) swapins, 0(0) swapouts. Networks: packets: 84178/42M in, 67523/12M out.
Disks: 245300/6196M read, 133444/4176M written.

PID  COMMAND      %CPU  TIME     #TH   #WQ  #PORT MEM    PURG  CMPRS  PGRP PPID STATE    BOOSTS          %CPU_ME %CPU_OTHRS UID  FAULTS
922  screencaptur 0.1   00:00.16 4     2    152   3712K  0B    0B     922  1    sleeping *0[1634+]       0.00000 0.04138    501  5498
921  screencaptur 1.2   00:01.13 2     1    54    3304K  636K  0B     308  308  sleeping *0[1]           0.04138 0.00000    501  112271+
920  java         392.1 01:03.51 41/8  1    127   682M+  0B    0B     453  453  running  *0[1]           0.00000 0.00000    501  187405+
919  java         0.0   00:03.06 29    1    103   144M   0B    0B     453  453  sleeping *0[1]           0.00000 0.00000    501  42976
918  java         0.0   00:01.45 33    1    111   74M    0B    0B     453  1    sleeping *0[1]           0.00000 0.00000    501  24572
917  ocspd        0.0   00:00.02 2     1    27    1340K  0B    0B     917  1    sleeping *0[1]           0.00000 0.00000    0    2131
916  java         0.0   00:09.66 31    1    107   198M   0B    0B     453  453  sleeping *0[1]           0.00000 0.00000    501  91106
911  mdworker_sha 0.0   00:00.04 3     1    59    3072K  0B    0B     911  1    sleeping *0[1]           0.00000 0.00000    501  3778
910  mdworker_sha 0.0   00:00.04 3     1    59    3112K  0B    0B     910  1    sleeping *0[1]           0.00000 0.00000    501  3784
```

- Whereas the sequential implementation does not:

```
Processes: 279 total, 3 running, 276 sleeping, 1098 threads                                                            19:25:59
Load Avg: 1.56, 1.58, 1.71  CPU usage: 25.80% user, 2.7% sys, 72.11% idle   SharedLibs: 288M resident, 63M data, 37M linkedit.
MemRegions: 25359 total, 2449M resident, 158M private, 446M shared. PhysMem: 7362M used (1424M wired), 829M unused.
VM: 1254G vsize, 1297M framework vsize, 0(0) swapins, 0(0) swapouts. Networks: packets: 22615/6950K in, 19320/4691K out.
Disks: 193532/4868M read, 91764/3399M written.

PID  COMMAND      %CPU  TIME     #TH   #WQ  #PORT MEM    PURG  CMPR PGRP PPID STATE    BOOSTS      %CPU_ME %CPU_OTHRS UID  FAULTS
666  screencaptur 0.0   00:00.31 3     1    148   3576K  0B    0B   666  1    sleeping *0[3113]    0.00000 0.00000    501  5515
665  screencaptur 0.0   00:02.71 2     1    56    3360K  636K  0B   308  308  sleeping *0[1]       0.00000 0.00000    501  276900
664  Python       99.5  00:41.89 1/1   0    10    15M    0B    0B   664  632  running  *0[1]       0.00000 0.00000    501  5093
650  com.apple.iC 0.0   00:00.05 2     1    55    3072K  0B    0B   650  1    sleeping  0[1]       0.00000 0.00000    501  4396
632  bash         0.0   00:00.04 1     0    21    892K   0B    0B   632  631  sleeping *0[1]       0.00000 0.00000    501  1524
631  login        0.0   00:00.01 2     1    31    1032K  0B    0B   631  488  sleeping *0[9]       0.00000 0.00000    0    1741
630  mdworker_sha 0.0   00:00.07 4     1    61    4580K  0B    0B   630  1    sleeping *0[1]       0.00000 0.00000    501  8055
624  trustd       0.0   00:00.09 2     2    37    4188K  168K  0B   624  1    sleeping  0[35]      0.00000 0.00000    89   3012
```

- SimulAkka thus has promise of horizontal scalability.

Systems used

- Sparkulation was run via Databricks.com hosting, which provides a cluster with a total of 6GB of memory - this ran all the code snippets that interacted with the underlying Spark framework.

Systems used

- Sparkulation was run via Databricks.com hosting, which provides a cluster with a total of 6GB of memory - this ran all the code snippets that interacted with the underlying Spark framework.

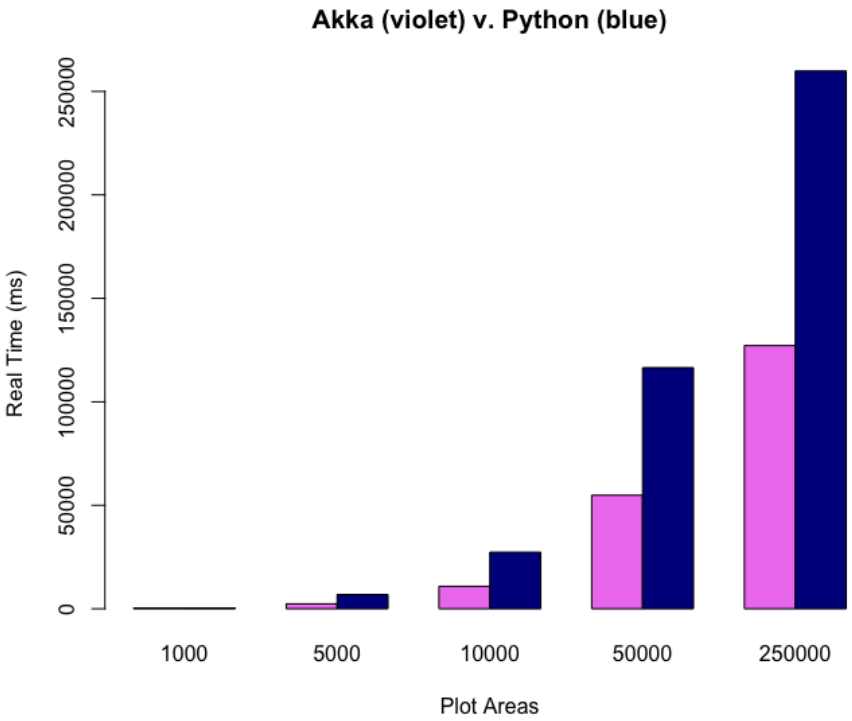
- SimulAkka was run on a system with a quad-core Intel i5 3.4 GHz processor and 8GB of memory - this ran both the SimulAkka-powered simulation, and a simpler (non-Actor model) Python implementation for comparison.

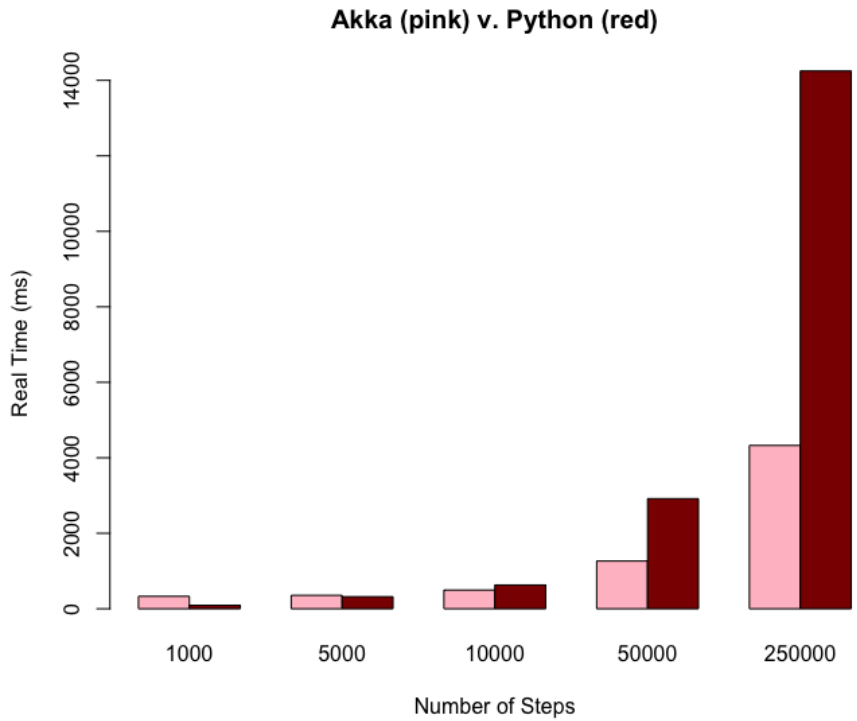
Overview

Sparkulation performance (versus R)

- Spark was 20% faster than R in my trial run of a single, relatively small (73mb) CSV that easily fits in most home computer's memory. Processing the entirety (21 gigabytes+) of FIADB data would give a significant advantage to Spark, both in feasibility and performance.

SimulAkka performance





Overview

```
void Plant() {
  //#pragma omp parallel for num_threads(NUM_THREADS)
  for (unsigned short s = 0; s < nS; s++) {
    cohort[K].s = s;
    //          if(OH==0) value[s].caTotal=FA/2;
    //          cohort[K].n = (species[s].n /(double)nP) *
    //(value[s].caTotal/FA);
    cohort[K].n = species[s].n / (double)nP;
    cohort[K].d = species[s].d;
    cohort[K].h = height(cohort[K]);
    cohort[K].r = radius(cohort[K]);
    K++;
    value[s].kTotal++;
    value[nS].kTotal++;
  }
}
...
```

- Automatically link the Spark pipeline with the Akka simulation system

- Automatically link the Spark pipeline with the Akka simulation system
- Fully Implement Sortie-PPA in SimulAkka

- Automatically link the Spark pipeline with the Akka simulation system
- Fully Implement Sortie-PPA in SimulAkka
- Explore the deployment of Spark on Amazon EMR to process over 20 million trees

- Automatically link the Spark pipeline with the Akka simulation system
- Fully Implement Sortie-PPA in SimulAkka
- Explore the deployment of Spark on Amazon EMR to process over 20 million trees
- Add soil biogeochemistry (C and N) to SimulAkka

- Automatically link the Spark pipeline with the Akka simulation system
- Fully Implement Sortie-PPA in SimulAkka
- Explore the deployment of Spark on Amazon EMR to process over 20 million trees
- Add soil biogeochemistry (C and N) to SimulAkka
- Port climate change models to SimulAkka and add bioclimatic data processing to Sparkulation

- Automatically link the Spark pipeline with the Akka simulation system
- Fully Implement Sortie-PPA in SimulAkka
- Explore the deployment of Spark on Amazon EMR to process over 20 million trees
- Add soil biogeochemistry (C and N) to SimulAkka
- Port climate change models to SimulAkka and add bioclimatic data processing to Sparkulation
- Once fast data analysis pipeline is ready, reparametrize and update models

- Automatically link the Spark pipeline with the Akka simulation system
- Fully Implement Sortie-PPA in SimulAkka
- Explore the deployment of Spark on Amazon EMR to process over 20 million trees
- Add soil biogeochemistry (C and N) to SimulAkka
- Port climate change models to SimulAkka and add bioclimatic data processing to Sparkulation
- Once fast data analysis pipeline is ready, reparametrize and update models
- Make attractive and healthy ecology codebase in order to encourage code reuse and maintenance by scientific community

Questions?