A dynamic optimization model for power and performance management of virtualized clusters

Vinicius Petrucci, Orlando Loques

Univ. Federal Fluminense

Niteroi, Rio de Janeiro, Brasil

Daniel Mossé

Univ. of Pittsburgh, USA

Impact of energy consumption

- Energy crisis
  - High energy costs and demand growing
- Environmental impacts
  - Carbon dioxide emission
  - Global warming
- Economic impacts
  - Resource scarcity leads to higher market prices
  - Clean and renewable sources may have high costs
- Power-efficiency is a fundamental concern in today's large server clusters / data-centers

Our solution

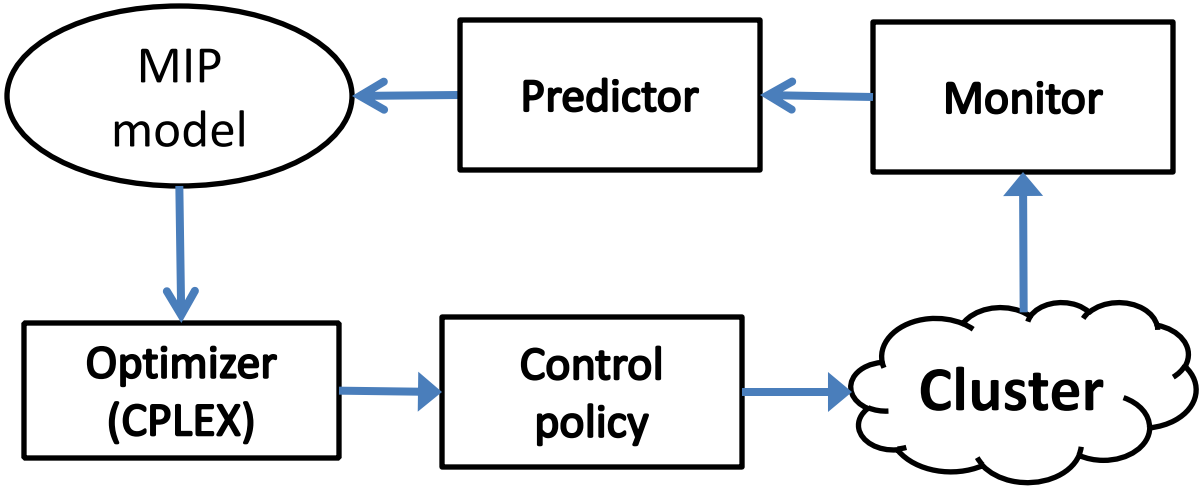
- Integrate power and performance management in heterogeneous server clusters
- Virtualized platform targeted at hosting multiple independent web applications
- Optimization approach to
  1. dynamically manage the cluster power consumption
  2. meet the application's performance demands

Optimization approach

- The optimization problem:
  - Determine the most power-efficient cluster configuration that can handle a given workload
  - Variant of the bin packing problem
- A cluster configuration is given by
  1. which servers must be active and their respective CPU frequencies
  2. a corresponding mapping of the apps (running on top of VMs) to physical servers

Optimization approach

- Optimization is given by a MIP model solved periodically in a control loop fashion
  1. Solve an optimization problem
  2. Use the solution to configure the cluster



Power and performance model

Server 1: ampere
CPU: AMD Athlon(tm) 64 X2 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	81.5	66.3	94.7
1800	101.8	70.5	168.4
2000	109.8	72.7	187.6

Server 2: coulomb
CPU: AMD Athlon(tm) 64 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	75.2	67.4	47.5
1800	89.0	70.9	84.6
2000	94.5	72.4	94.0
2200	100.9	73.8	102.8
2400	107.7	75.2	111.5

Server 3: hertz
CPU: AMD Athlon(tm) 64 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	71.6	63.9	47.0
1800	85.5	67.2	83.9
2000	90.7	68.7	93.0
2200	96.5	69.9	102.3
2400	103.2	71.6	110.5

Server 4: joule
CPU: AMD Athlon(tm) 64 3500+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	74.7	66.6	47.1
1800	95.7	73.8	84.0
2000	103.1	76.9	93.5
2200	110.6	80.0	102.0

Server 5: ohm
CPU: AMD Athlon(tm) 64 X2 5000+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	82.5	65.8	92.9
1800	99.2	68.5	165.9
2000	107.3	70.6	184.4
2200	116.6	72.3	201.0
2400	127.2	74.3	218.1
2600	140.1	76.9	235.3

Different power and performance levels in heterogeneous server clusters

Power and performance model

Server 1: ampere
CPU: AMD Athlon(tm) 64 X2 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	81.5	66.3	94.7
1800	101.8	70.5	168.4
2000	109.8	72.7	187.6

Server 2: coulomb
CPU: AMD Athlon(tm) 64 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	75.2	67.4	47.5
1800	89.0	70.9	84.6
2000	94.5	72.4	94.0
2200	100.9	73.8	102.8
2400	107.7	75.2	111.5

Server 3: hertz
CPU: AMD Athlon(tm) 64 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	71.6	63.9	47.0
1800	85.5	67.2	83.9
2000	90.7	68.7	93.0
2200	96.5	69.9	102.3
2400	103.2	71.6	110.5

Server 4: joule
CPU: AMD Athlon(tm) 64 3500+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	74.7	66.6	47.1
1800	95.7	73.8	84.0
2000	103.1	76.9	93.5
2200	110.6	80.0	102.0

Server 5: ohm
CPU: AMD Athlon(tm) 64 X2 5000+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	82.5	65.8	92.9
1800	99.2	68.5	165.9
2000	107.3	70.6	184.4
2200	116.6	72.3	201.0
2400	127.2	74.3	218.1
2600	140.1	76.9	235.3

Dynamic Voltage/Frequency Scaling

Power and performance model

Server 1: ampere
CPU: AMD Athlon(tm) 64 X2 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	91.5	66.3	94.7
1800	101.8	70.5	168.4
2000	109.8	72.7	187.6

Server 2: coulomb
CPU: AMD Athlon(tm) 64 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	75.2	67.4	47.5
1800	89.0	70.9	84.6
2000	94.5	72.4	94.0
2200	100.9	73.8	102.8
2400	107.7	75.2	111.5

Server 3: hertz
CPU: AMD Athlon(tm) 64 3800+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	71.6	63.9	47.0
1800	85.5	67.2	83.9
2000	90.7	68.7	93.0
2200	96.5	69.9	102.3
2400	103.2	71.6	110.5

Server 4: joule
CPU: AMD Athlon(tm) 64 3500+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	74.7	66.6	47.1
1800	95.7	73.8	84.0
2000	103.1	76.9	93.5
2200	110.6	80.0	102.0

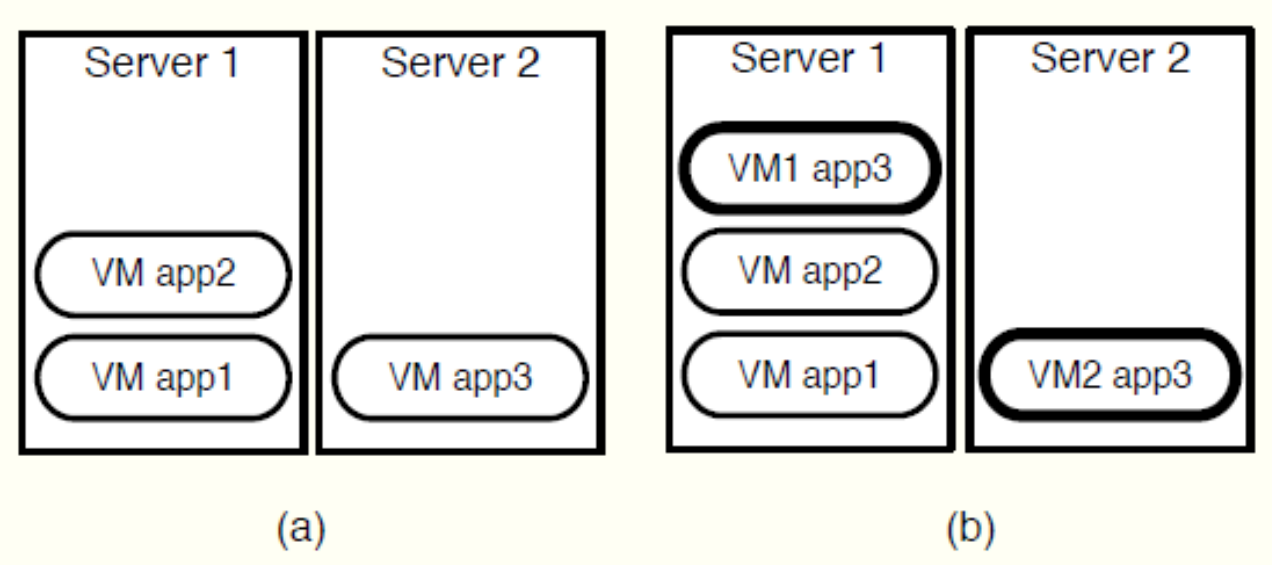
Server 5: ohm
CPU: AMD Athlon(tm) 64 X2 5000+

Freq. (MHz)	P_{busy} (W)	P_{idle} (W)	Perf. (Req/s)
1000	82.5	65.8	92.9
1800	99.2	68.5	165.9
2000	107.3	70.6	184.4
2200	116.6	72.3	201.0
2400	127.2	74.3	218.1
2600	140.1	76.9	235.3

Server on/off mechanisms (e.g., standby + Wake-on-LAN)

Optimization model

Two ways of mapping app workloads to VMs



(a) every app runs in only one VM instance on a given server,
(b) one given app may run in more than one VM instance, whereas these VMs are balanced among multiple servers

Optimization model

Input variables:

N = set of servers in the cluster

F_i = set of frequencies for each server *i* in N

M = set of apps/services in the cluster

cap_{ij} = capacity (e.g., req/s) of server *i* at frequency *j*

pb_{ij} , pi_{ij} = power costs (idle and busy) of server *i* at freq. *j*

d_k = demand (workload) of app *k*

Decision variables:

y_{ij} = 1 if server *i* runs at frequency *j*, 0 otherwise

x_{ijk} = 1 if server *i* uses frequency *j* to run app *k*, 0 otherwise

α_{ijk} = utilization variable [0,1] of server *i* at frequency *j*

Optimization model

Problem formulation

$$\textit{Minimize} \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \qquad (1)$$

$$\textit{Subject to} \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N,\ \forall j \in F_i \ (2)$$

$$\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \qquad \forall k \in M \ (3)$$

$$\sum_{j \in F_i} y_{ij} \leq 1 \qquad \forall i \in N \ (4)$$

$$\alpha_{ij} \leq y_{ij} \qquad \forall i \in N,\ \forall j \in F_i \ (5)$$

$$x_{ijk} \in \{0, 1\},\ y_{ij} \in \{0, 1\},\ \alpha_{ij} \in [0, 1]$$

Optimization model

Minimization of the overall power consumption

$$Minimize \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \qquad (1)$$

$$Subject\ to \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N,\ \forall j \in F_i \ (2)$$

$$\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \qquad \forall k \in M \ (3)$$

$$\sum_{j \in F_i} y_{ij} \leq 1 \qquad \forall i \in N \ (4)$$

$$\alpha_{ij} \leq y_{ij} \qquad \forall i \in N,\ \forall j \in F_i \ (5)$$

$$x_{ijk} \in \{0, 1\},\ y_{ij} \in \{0, 1\},\ \alpha_{ij} \in [0, 1]$$

Optimization model

Server capacity constraints

$$\textit{Minimize} \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \qquad (1)$$

$$\textit{Subject to} \quad \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N,\ \forall j \in F_i \qquad (2)$$

$$\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \qquad \forall k \in M \qquad (3)$$

$$\sum_{j \in F_i} y_{ij} \leq 1 \qquad \forall i \in N \qquad (4)$$

$$\alpha_{ij} \leq y_{ij} \qquad \forall i \in N,\ \forall j \in F_i \qquad (5)$$

$$x_{ijk} \in \{0, 1\},\ y_{ij} \in \{0, 1\},\ \alpha_{ij} \in [0, 1]$$

Optimization model

Application allocation constraints

$$\textit{Minimize} \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \qquad (1)$$

$$\textit{Subject to} \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N,\ \forall j \in F_i \ (2)$$

$$\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \qquad \forall k \in M \ (3)$$

$$\sum_{j \in F_i} y_{ij} \leq 1 \qquad \forall i \in N \ (4)$$

$$\alpha_{ij} \leq y_{ij} \qquad \forall i \in N,\ \forall j \in F_i \ (5)$$

$$x_{ijk} \in \{0, 1\},\ y_{ij} \in \{0, 1\},\ \alpha_{ij} \in [0, 1]$$

Optimization model

Server frequency selection constraints

$$Minimize \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \qquad (1)$$

$$Subject\ to \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N,\ \forall j \in F_i \ (2)$$

$$\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \qquad \forall k \in M \ (3)$$

$$\sum_{j \in F_i} y_{ij} \leq 1 \qquad \forall i \in N \ (4)$$

$$\alpha_{ij} \leq y_{ij} \qquad \forall i \in N,\ \forall j \in F_i \ (5)$$

$$x_{ijk} \in \{0, 1\},\ y_{ij} \in \{0, 1\},\ \alpha_{ij} \in [0, 1]$$

Optimization model

Bound constraints for utilization and server selection variables

$$\textit{Minimize} \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \qquad (1)$$

$$\textit{Subject to} \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N,\ \forall j \in F_i \ (2)$$

$$\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \qquad \forall k \in M \ (3)$$

$$\sum_{j \in F_i} y_{ij} \leq 1 \qquad \forall i \in N \ (4)$$

$$\alpha_{ij} \leq y_{ij} \qquad \forall i \in N,\ \forall j \in F_i \ (5)$$

$$x_{ijk} \in \{0, 1\},\ y_{ij} \in \{0, 1\},\ \alpha_{ij} \in [0, 1]$$

Optimization model

Domains of the decision variables

$$\textit{Minimize} \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \qquad (1)$$

$$\textit{Subject to} \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N,\ \forall j \in F_i \ (2)$$

$$\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \qquad \forall k \in M \ (3)$$

$$\sum_{j \in F_i} y_{ij} \leq 1 \qquad \forall i \in N \ (4)$$

$$\alpha_{ij} \leq y_{ij} \qquad \forall i \in N,\ \forall j \in F_i \ (5)$$

$$x_{ijk} \in \{0, 1\},\ y_{ij} \in \{0, 1\},\ \alpha_{ij} \in [0, 1]$$

Extensions to the model

- Application workload balancing
  - Represent fractions of an application workload in a given selected server
  - Relax the allocation variable (to be a real domain)
- Server switching on/off and VM migration costs
  - Keep state from previous configuration
  - Add penalty to the objective function

Optimization control loop

```
do periodically:
  // 1. Input variables
  d = getDemandVector()
  curUsage = getCurrentUsage()
  curAlloc = getCurrentAlloc()

  // 2. Run optimization
  newUsage, newAlloc = bestConfig(d)

  // 3. Generate usage and alloc sets for changes
  chgUsage = sort(diff(newUsage, curUsage))
  chgAlloc = sort(diff(newAlloc, curAlloc))

  // 4. Power management operations
  for (i, j) in chgUsage:
      if j == 0:
          turnOff(i)
      else:
          if curUsage[i] == 0:
              turnOn(i)
          setFreq(i, j)
```

```
// 5. Virtualization management operations
for (k, i) in chgAlloc:
    if i == 0:
        stopVm(k, curAlloc[k])
    else:
        if curAlloc[k] == 0:
            startVm(k, i)
        else:
            migrateVm(k, curAlloc[k], i)
```

Optimization control loop

```
do periodically:
  // 1. Input variables
  d = getDemandVector()
  curUsage = getCurrentUsage()
  curAlloc = getCurrentAlloc()

  // 2. Run optimization
  newUsage, newAlloc = bestConfig(d)

  // 3. Generate usage and alloc sets for changes
  chgUsage = sort(diff(newUsage, curUsage))
  chgAlloc = sort(diff(newAlloc, curAlloc))

  // 4. Power management operations
  for (i, j) in chgUsage:
      if j == 0:
          turnOff(i)
      else:
          if curUsage[i] == 0:
              turnOn(i)
          setFreq(i, j)
```

```
// 5. Virtualization management operations
for (k, i) in chgAlloc:
    if i == 0:
        stopVm(k, curAlloc[k])
    else:
        if curAlloc[k] == 0:
            startVm(k, i)
        else:
            migrateVm(k, curAlloc[k], i)
```

Control loop steps:

(1) Collect and store the most recent values of the optimization input variables;

(2) Construct and solve a new optimization problem instance, yielding a new optimal configuration;

(3) Apply the changes in the system, transitioning the system to a new state given by the new optimized configuration.

Optimization control loop

```
do periodically:
  // 1. Input variables
  d = getDemandVector()
  curUsage = getCurrentUsage()
  curAlloc = getCurrentAlloc()

  // 2. Run optimization
  newUsage, newAlloc = bestConfig(d)

  // 3. Generate usage and alloc sets for changes
  chgUsage = sort(diff(newUsage, curUsage))
  chgAlloc = sort(diff(newAlloc, curAlloc))

  // 4. Power management operations
  for (i, j) in chgUsage:
      if j == 0:
          turnOff(i)
      else:
          if curUsage[i] == 0:
              turnOn(i)
          setFreq(i, j)
```

```
// 5. Virtualization management operations
for (k, i) in chgAlloc:
    if i == 0:
        stopVm(k, curAlloc[k])
    else:
        if curAlloc[k] == 0:
            startVm(k, i)
        else:
            migrateVm(k, curAlloc[k], i)
```

Control loop steps:

(1) Collect and store the most recent values of the optimization input variables;

(2) Construct and solve a new optimization problem instance, yielding a new optimal configuration;

(3) Apply the changes in the system, transitioning the system to a new state given by the new optimized configuration.

Optimization control loop

```
do periodically:
  // 1. Input variables
  d = getDemandVector()
  curUsage = getCurrentUsage()
  curAlloc = getCurrentAlloc()

  // 2. Run optimization
  newUsage, newAlloc = bestConfig(d)

  // 3. Generate usage and alloc sets for changes
  chgUsage = sort(diff(newUsage, curUsage))
  chgAlloc = sort(diff(newAlloc, curAlloc))
```

```
// 5. Virtualization management operations
for (k, i) in chgAlloc:
    if i == 0:
        stopVm(k, curAlloc[k])
    else:
        if curAlloc[k] == 0:
            startVm(k, i)
        else:
            migrateVm(k, curAlloc[k], i)
```

```
// 4. Power management operations
for (i, j) in chgUsage:
    if j == 0:
        turnOff(i)
    else:
        if curUsage[i] == 0:
            turnOn(i)
        setFreq(i, j)
```

Control loop steps:

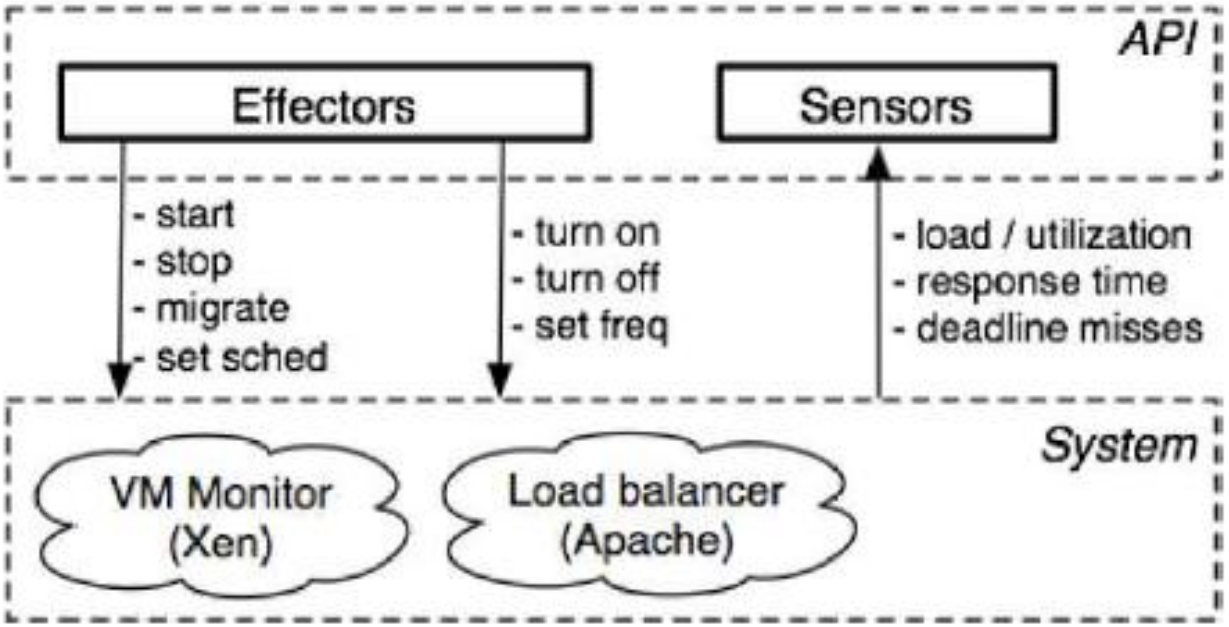
(1) Collect and store the most recent values of the optimization input variables;

(2) Construct and solve a new optimization problem instance, yielding a new optimal configuration;

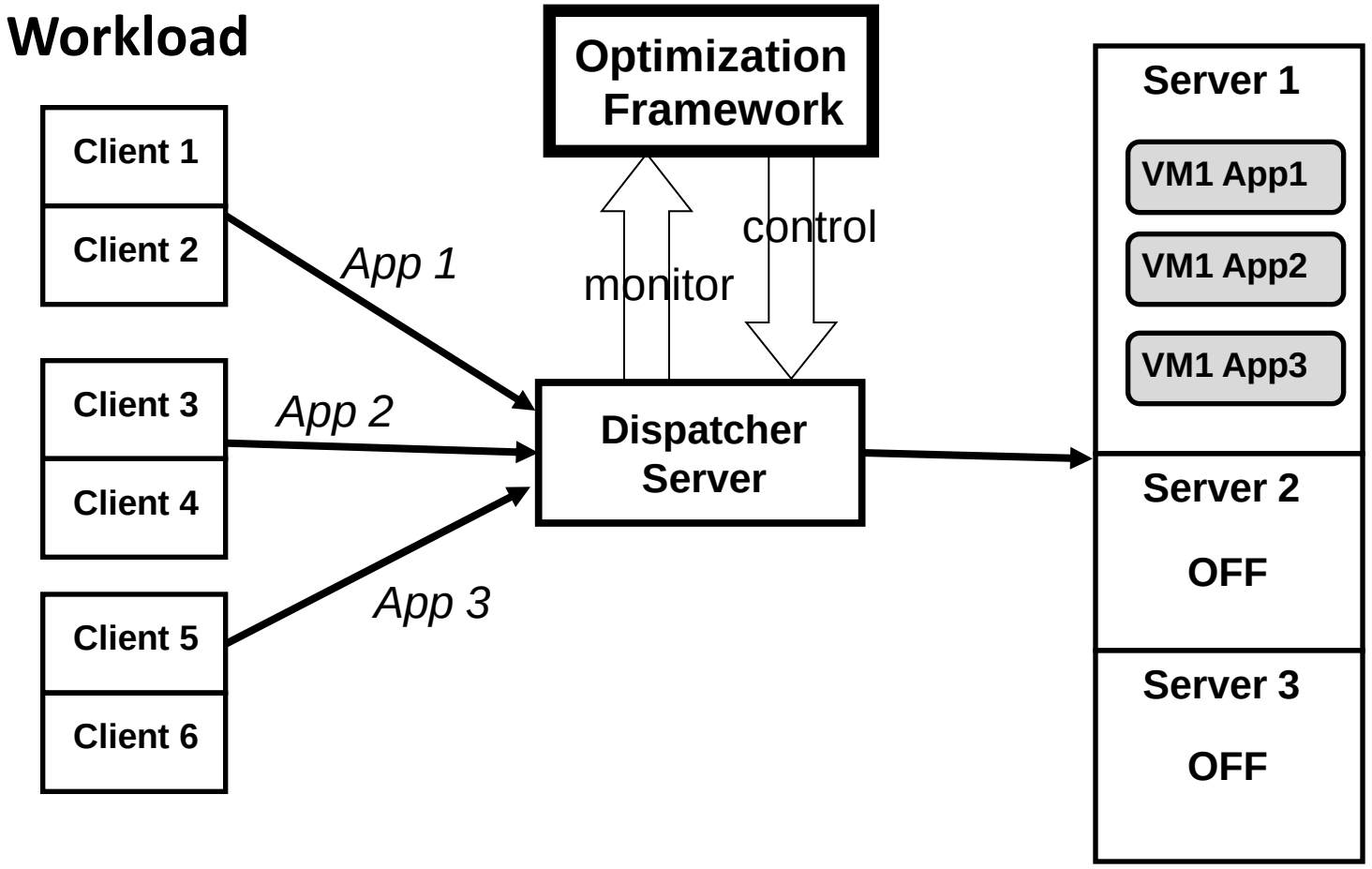
(3) Apply the changes in the system, transitioning the system to a new state given by the new optimized configuration.

Configuration support

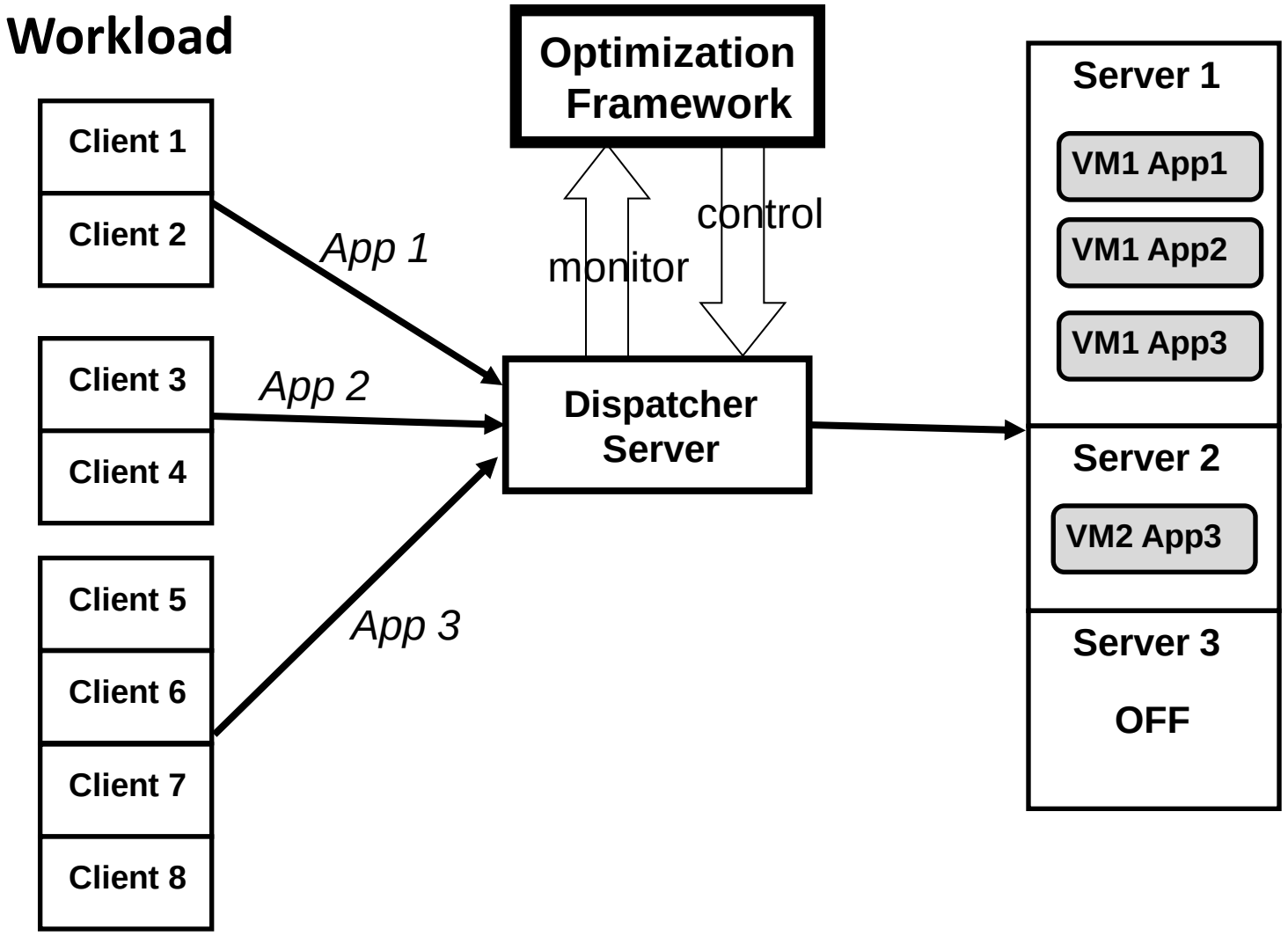
The optimization proposal relies on monitoring and configuration capabilities, such as VM migration and server on/off, which are described by means of an API [Petrucci et. al ACM SAC'09]



Server cluster architecture



Server cluster architecture



Evaluation

- Three distinct app workloads *based on* WC98
- Cluster setup with 5 physical servers
- Optimization model implemented using the CPLEX 11 package solver
- Power/performance benchmark for the servers
  - Workload generator: httperf tool
  - Power monitor: LabView with USB DAQ

Workload traces

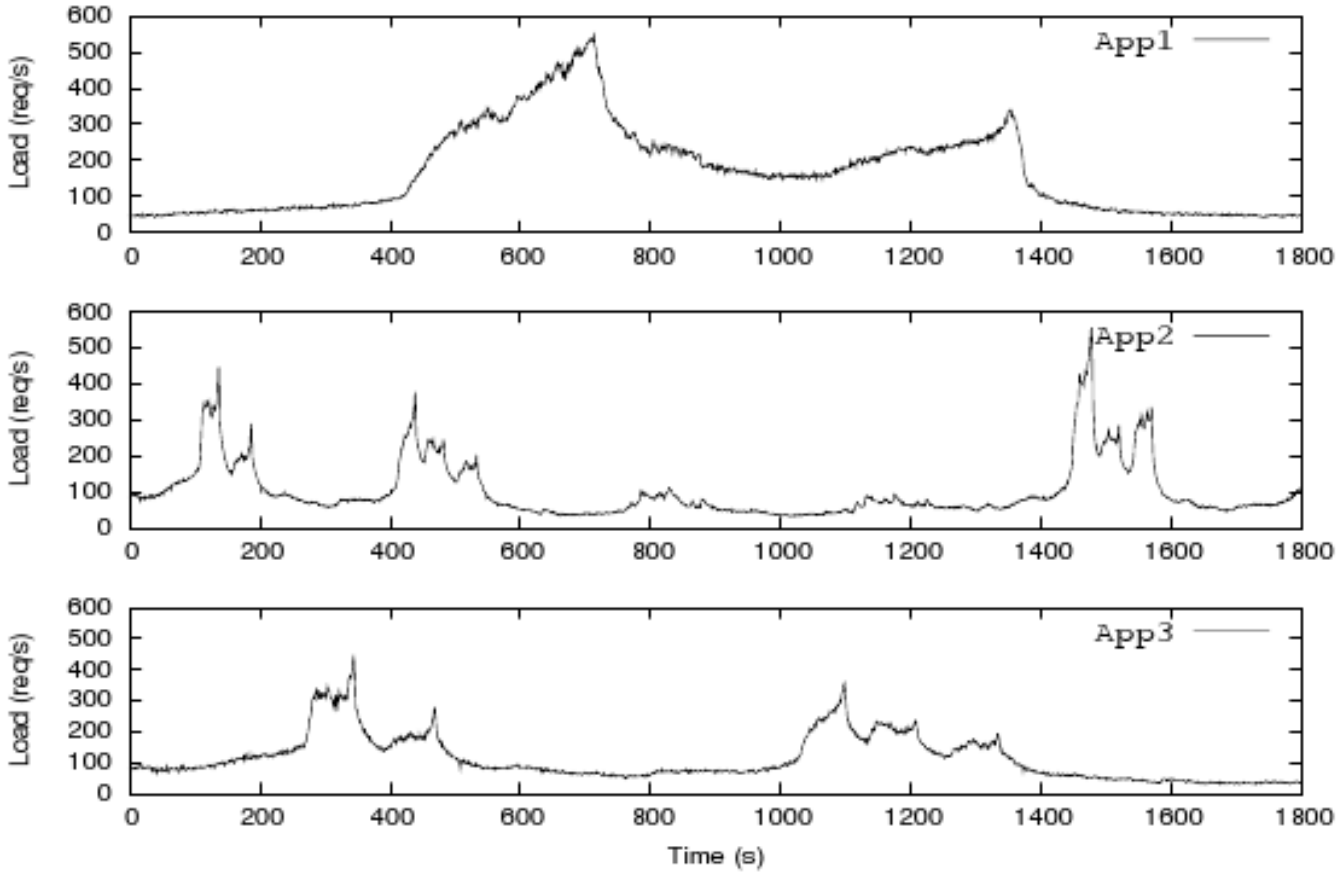


Figure 1: Workload traces for three different applications using HTTP logs from WC98

Optimization execution

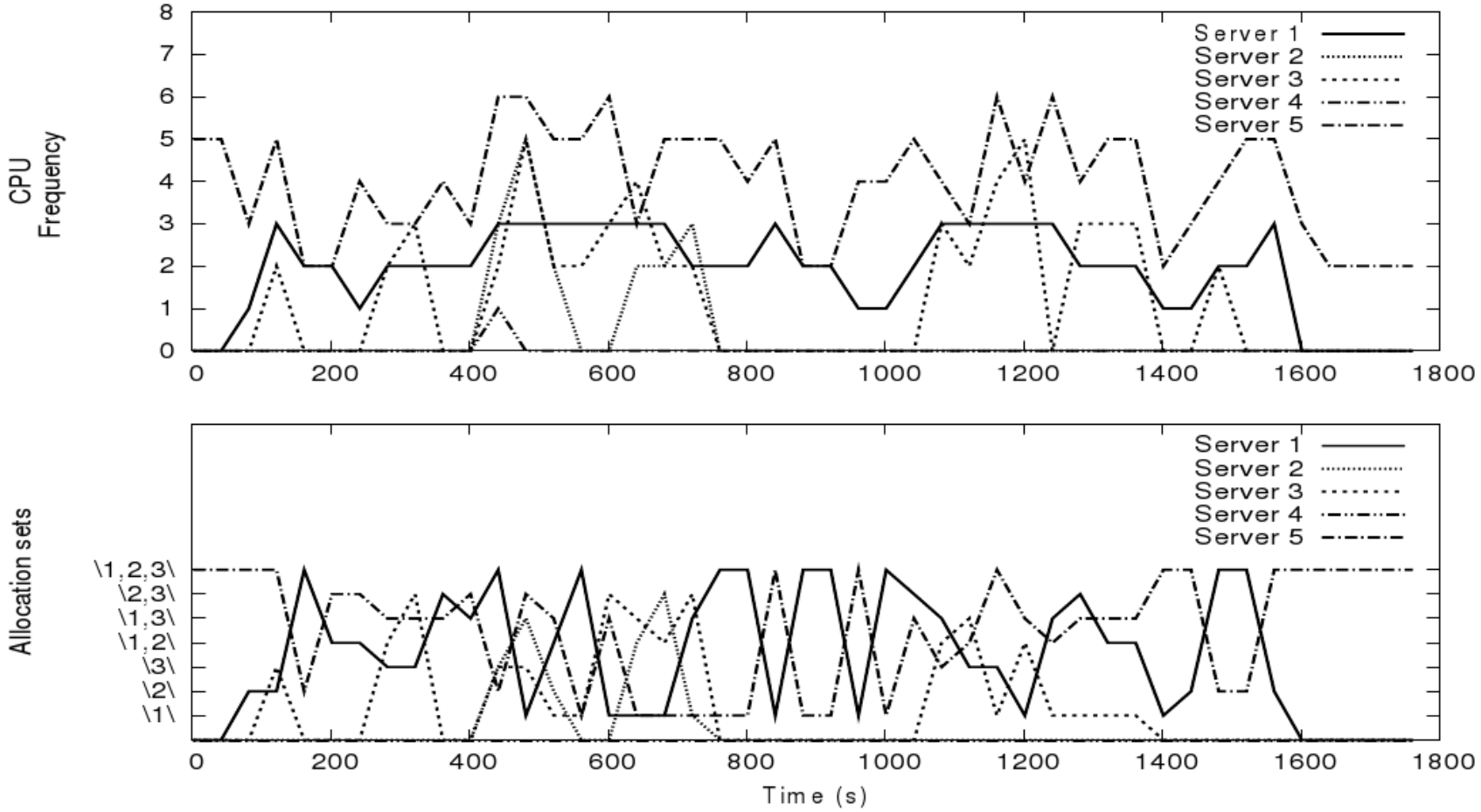


Figure 2: Dynamic optimization execution

Power consumption

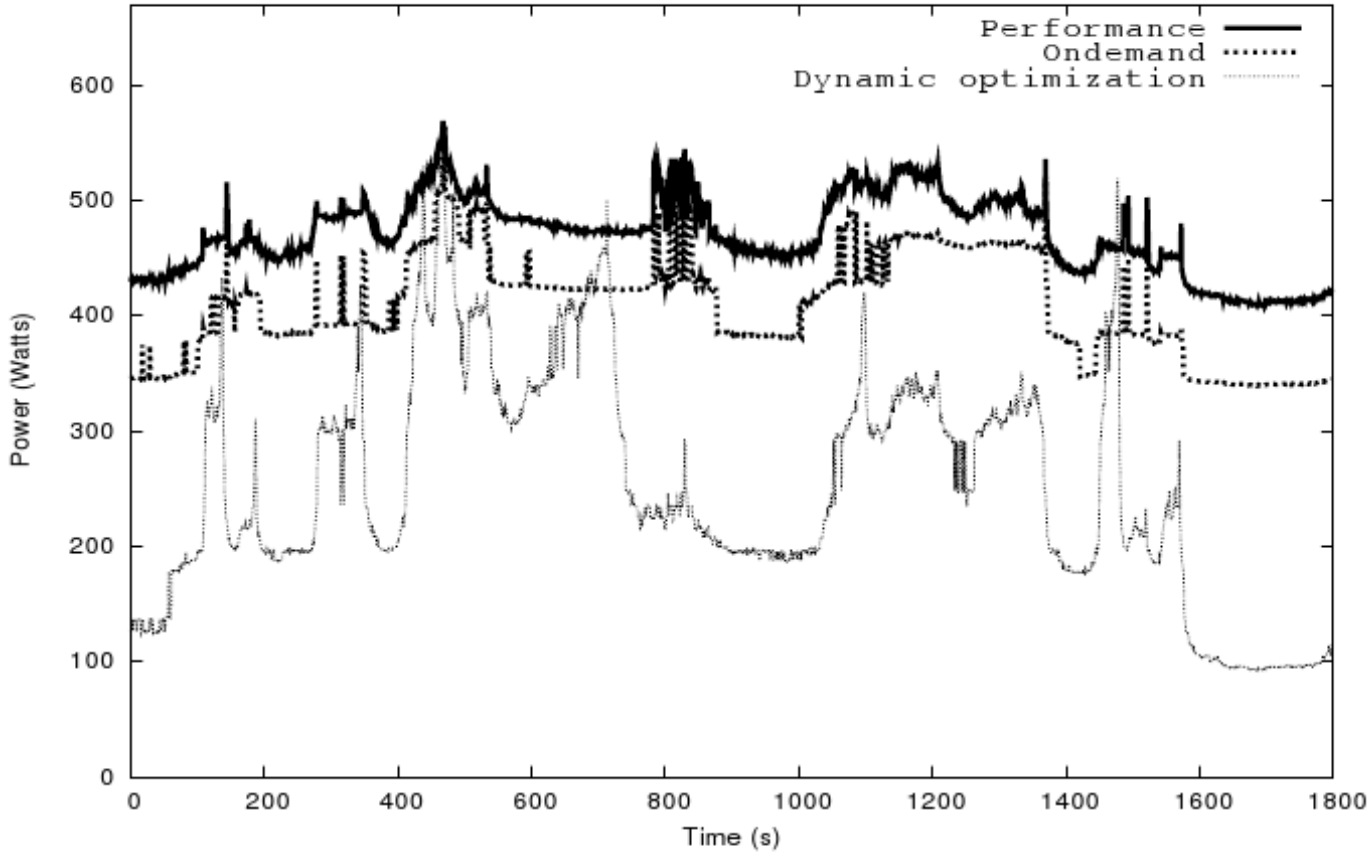


Figure 3: Cluster power consumption

Power consumption

- Comparison with Linux kernel CPU governors
- Energy consumption estimation

$$E = \sum_t \sum_{i,j} \alpha_{ij}^t \cdot pb_{ij} + (1 - \alpha_{ij}^t) \cdot pi_{ij}$$

i is an active server (and **j** is its operating frequency)
alpha is utilization at time **t** (**pb** and **pi** are idle and busy power)

Performance governor → $847{,}778.82\text{J} = 235.49\text{Wh}$
On-demand governor → $735{,}630.05\text{J} = 204.34\text{Wh}$
Optimization approach → $452{,}050.15\text{J} = 125.57\text{Wh}$

47% compared to `performance`

38% compared to `ondemand`

Scalability simulation

- CPLEX with time limit of 180 seconds (related to a given control period)

Server-App	Avg. (s)	Stdev. (s)	Max. (s)
(5,3)	0.022	0.018	0.070
(10,6)	0.054	0.035	0.250
(15,9)	0.062	0.038	0.240
(30,18)	0.392	0.913	8.610
(50,30)	13.630	29.959	180.010
(80,48)	58.941	53.570	180.020
(100,60)	80.135	52.394	180.030

- 180 executions (1800s of workload duration spaced by 10s) – one execution for each second

Scalability simulation

- Now including an optimality tolerance of 5% (180 executions)

Server-App	Avg. (s)	Stdev. (s)	Max. (s)
(5,3)	0.006	0.008	0.040
(10,6)	0.023	0.022	0.100
(15,9)	0.031	0.030	0.130
(30,18)	0.062	0.067	0.540
(50,30)	0.139	0.281	2.390
(80,48)	0.267	0.235	3.000
(100,60)	0.481	0.409	3.080
(200,120)	2.893	1.993	11.550
(350,210)	16.488	12.979	75.440
(500,300)	48.409	41.472	181.030

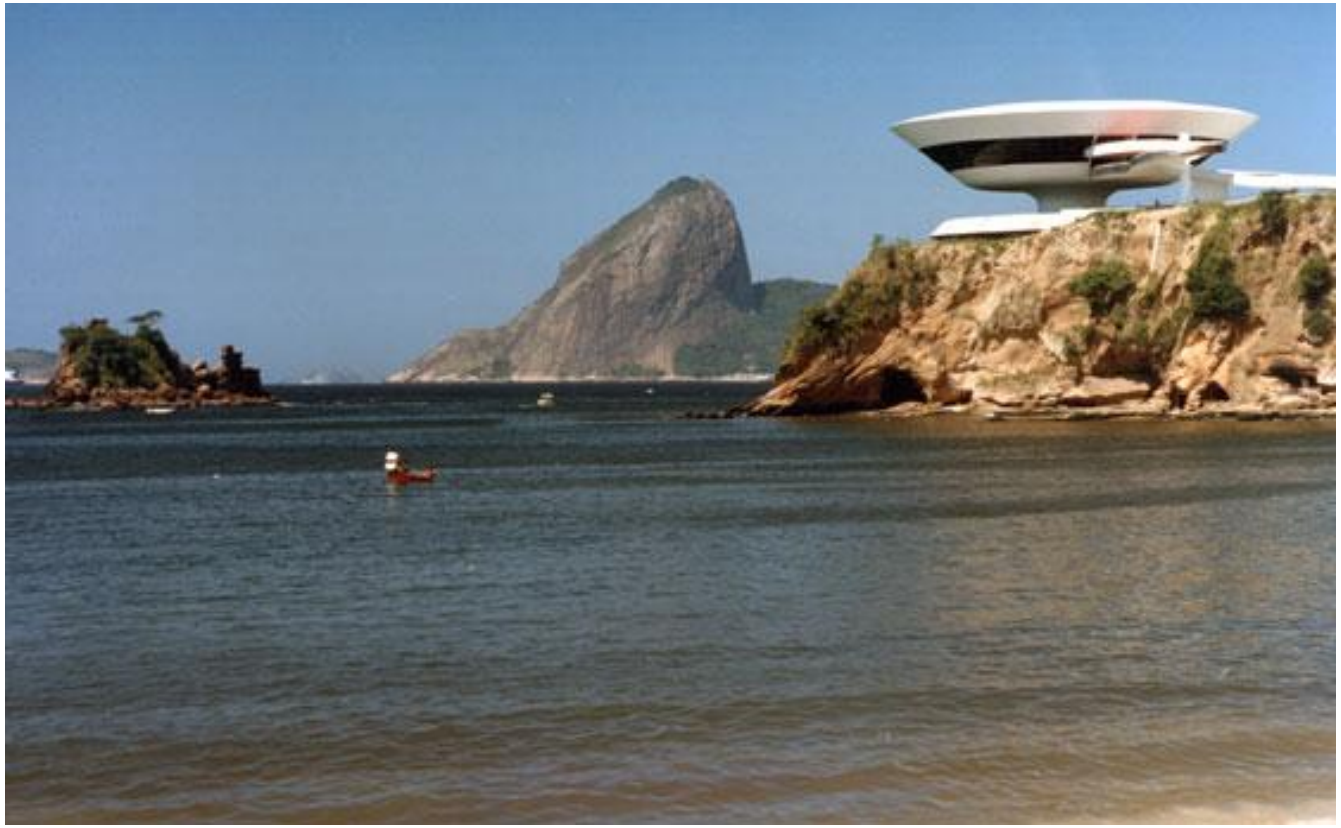
Conclusion

- We proposed an optimization solution for power and performance management
  - Targeted for virtualized server clusters
- The proposal includes an optimization model and a control loop strategy
- Simulations showed practicability and attractive power reductions compared to Linux governors
  - Mainly because of server on/off mechanisms

Current work

- Experimental evaluation in a real cluster test-bed
  - Optimization control loop implementation
  - Xen hypervisor and Apache web servers
- Analysis of overhead/cost in imposing dynamic cluster configurations
  - E.g., VM deployment/replication, live migration
- Improvements in the optimization decisions by leveraging predictive information about the workload
  - Well-known techniques for load forecasting
- Acceleration of the optimization process (branch-and-bound)
  - Problem-specific heuristics (upper bound) input for CPLEX
  - Valid inequalities to improve dual solution limits (lower bounds) of the MIP model

Thank you!



The contemporary Art museum in Niteroi, Rio de Janeiro

Variable-sized bin packing problem