Performance Scaling Variability and Energy Analysis for a Resilient ULFM-based PDE Solver

F. Rizzi[†], K. Morris[†], B. Cook[*], K. Sargsyan[†], P. Mycek[‡], O. LeMaitre[‡], O. Knio[‡], K. Dahlgren[†], B. Debusschere[†]

[†] Sandia National Laboratories, Livermore, CA, USA
[‡] Duke University, Durham, NC, USA
[*] LBNL, Berkeley, CA, USA

ScalA 2016, Salt Lake City
– Nov. 2016 –

Supported by the US Department of Energy (DOE)
Advanced Scientific Computing Research (ASCR)

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Do We Need Resilience?

Resilience:

Collection of techniques to keep applications running to a correct solution in a timely and efficient manner *despite* underlying system faults.

Addressing Failures in Exascale Computing, Snir *et al.*, 2012.

Toward Exascale Resilience, Cappello *et al.*, 2014.

Do We Need Resilience?

Resilience

Collection of techniques to keep applications running to a correct solution in a timely and efficient manner *despite* underlying system faults.

Addressing Failures in Exascale Computing, Snir *et al.*, 2012.

Toward Exascale Resilience, Cappello *et al.*, 2014.

IEEE, Feb.2016, Al Geist (ORNL)

"As a child, were you ever afraid that a monster lurking in your bedroom would leap out of the dark and get you?"

"Lack of resilience is a similar monster, hiding in the steel cabinets of the supercomputers and threatening to crash the largest computing machines."

Faults: Broadly Speaking

- Hard: activation is systematically reproducible.
- Soft: activation is not systematically reproducible.
- Active: fault causes an error.
- Dormant: fault does not cause an error.
  The dormant fault is activated when it causes an error.
- Permanent: presence is continuous in time.
- Transient: presence is temporary.
- Intermittent: fault is transient and reappears.

Addressing Failures in Exascale Computing, Snir *et al.*, 2012.

Challenges & Needs

- *Hypothetically* hardware can take care of most of it...
- ...at the expense of energy consumption, money, and asynchrony.
- Power is a big obstacle towards exascale.
- High tradeoffs between power, resiliency and performance.

  For instance: if an application can tolerate memory bit flips for certain parts of its memory, it can ask the OS to turn off ECC checks for those memory regions and potentially improve power and performance.
- Cross-cutting research needed to explore these areas.

The Lorenz attractor

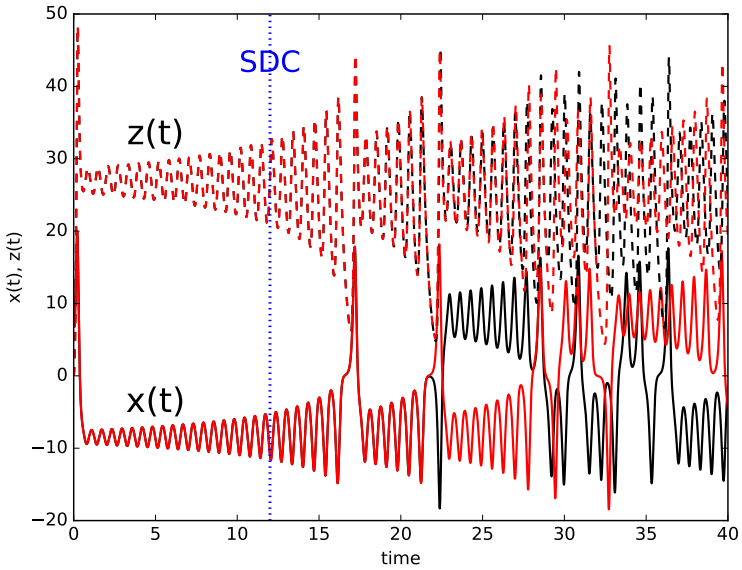
- Ed N. Lorenz, meterologist (1963): simplified model of convection in the earths atmosphere (also found in models of lasers and dynamos).
- System of coupled non-linear differential equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x), \qquad \frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y, \qquad \frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z$$

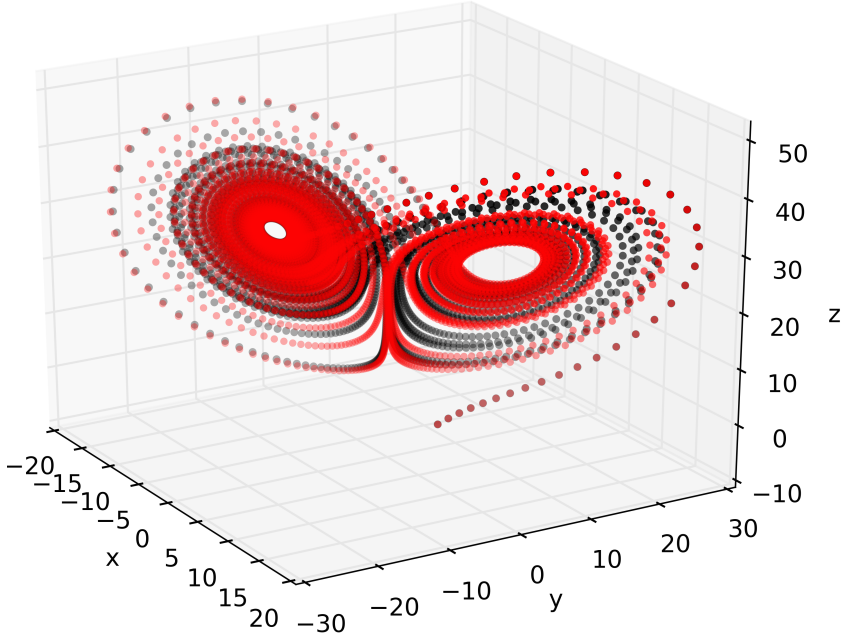
- Lorenz used $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$.
- Systems exhibits chaotic behavior for these (and nearby) values.
- Chaotic means highly sensitive to initial conditions.
  Small differences in initial conditions (such as those due to rounding errors in numerical computation) yield widely diverging outcomes for such dynamical systems, making long-term prediction impossible.

The Lorenz attractor: effect of Silent Data Corruption (SDCs)

- SDC: data corruption not raising errors/warnings.
- Start from $x_0 = 8$, $y_0 = 5$, $z_0 = 1$. Solve over $0 \leq t \leq 40$.
- Simulate SDC by randomly selecting a bit to flip in x at $t = 12$.
- 25*th* bit, *before* $= -11.02263241724513$, *after* $= -11.02312069849513$.
- Small change, but completely different solution for this "simple" test!



nominal, **corrupted**



nominal, **corrupted**

Algorithm

Resilient EXtreme Scale Scientific Simulations (REXSSS)

What?

Domain-decomposition-based preconditioner for PDEs.

- Currently for elliptic equations (1D, 2D).
- Other PDEs (in progress).

How?

Recasting the original PDE problem as a sampling problem, followed by a resilient data manipulation to achieve the final solution update.

Why?

- We do not characterize all types of system faults that can occur, but focus solely on the information that a simulation provides.
- Target: silent data corruptions (SDCs) and nodes/cores failing.

Algorithm Overview

1D problem:

$$\begin{cases} \mathcal{L}y(x) = g(x), & \text{in } \Omega = (x_-, x_+) \\ y(x_-) = y^-, \\ y(x_+) = y^+, \end{cases}$$

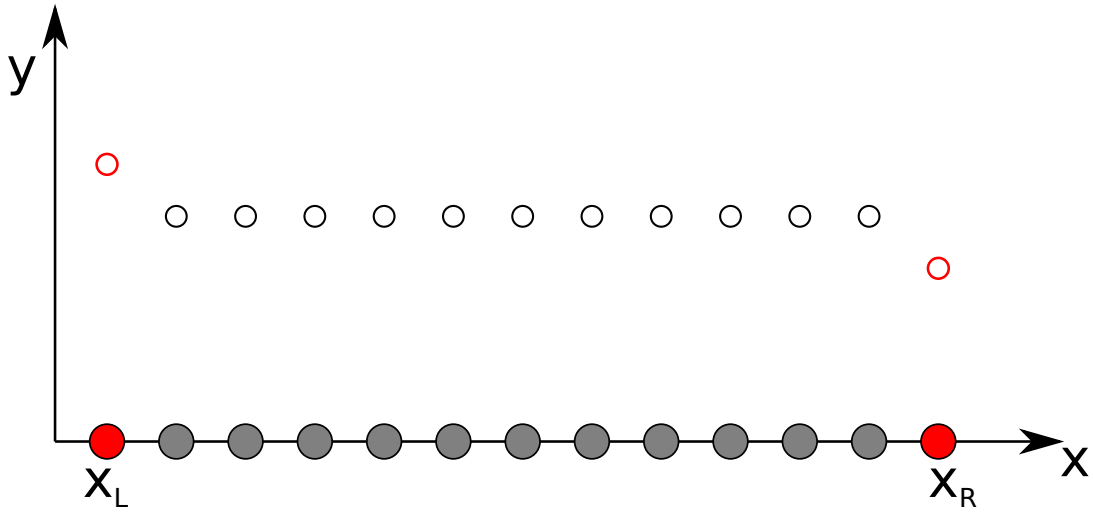
$\mathcal{L}$ is a **linear**, elliptic operator.

The solution at point x_0 **linearly depends** on the boundary conditions:

$$y(x_0) = a + by^- + cy^+.$$

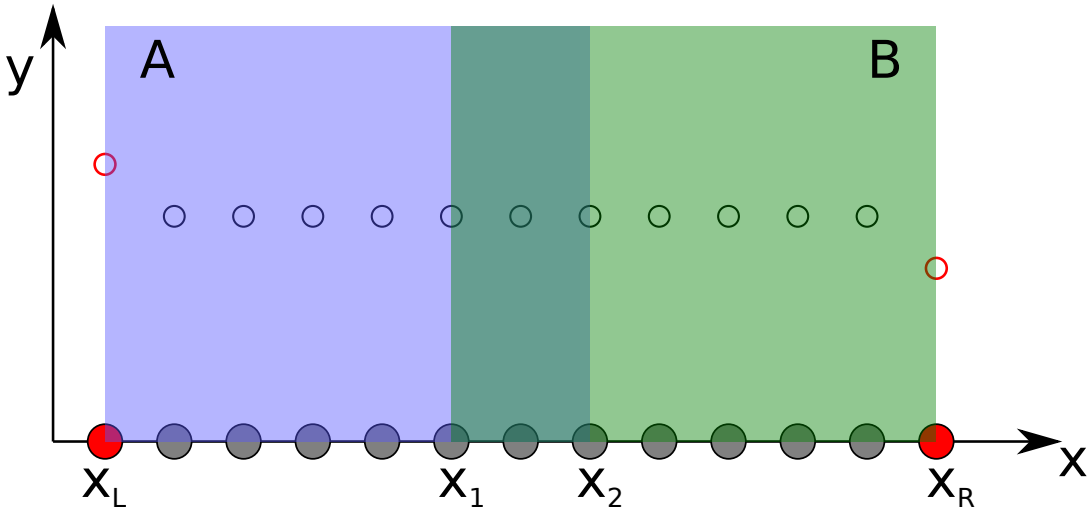
Algorithm Overview: Domain Decomposition & Boundary Maps

- Grid with current state.
- Partition space with overlapping subdomains.
- Treat subdomains independently.
- Define sampling range for each boundary. Sample and solve PDE locally.
- From samples build maps:
  $y_1 = f(y_2, y_L) = a + by_2 + cy_L$
  $y_2 = g(y_1, y_R) = d + ey_1 + fy_R$
  Maps link the subdomains.
- y_L, y_R are known BC.
- Solve, new state: (y_1^*, y_2^*).
- If needed: update range, repeat loop.



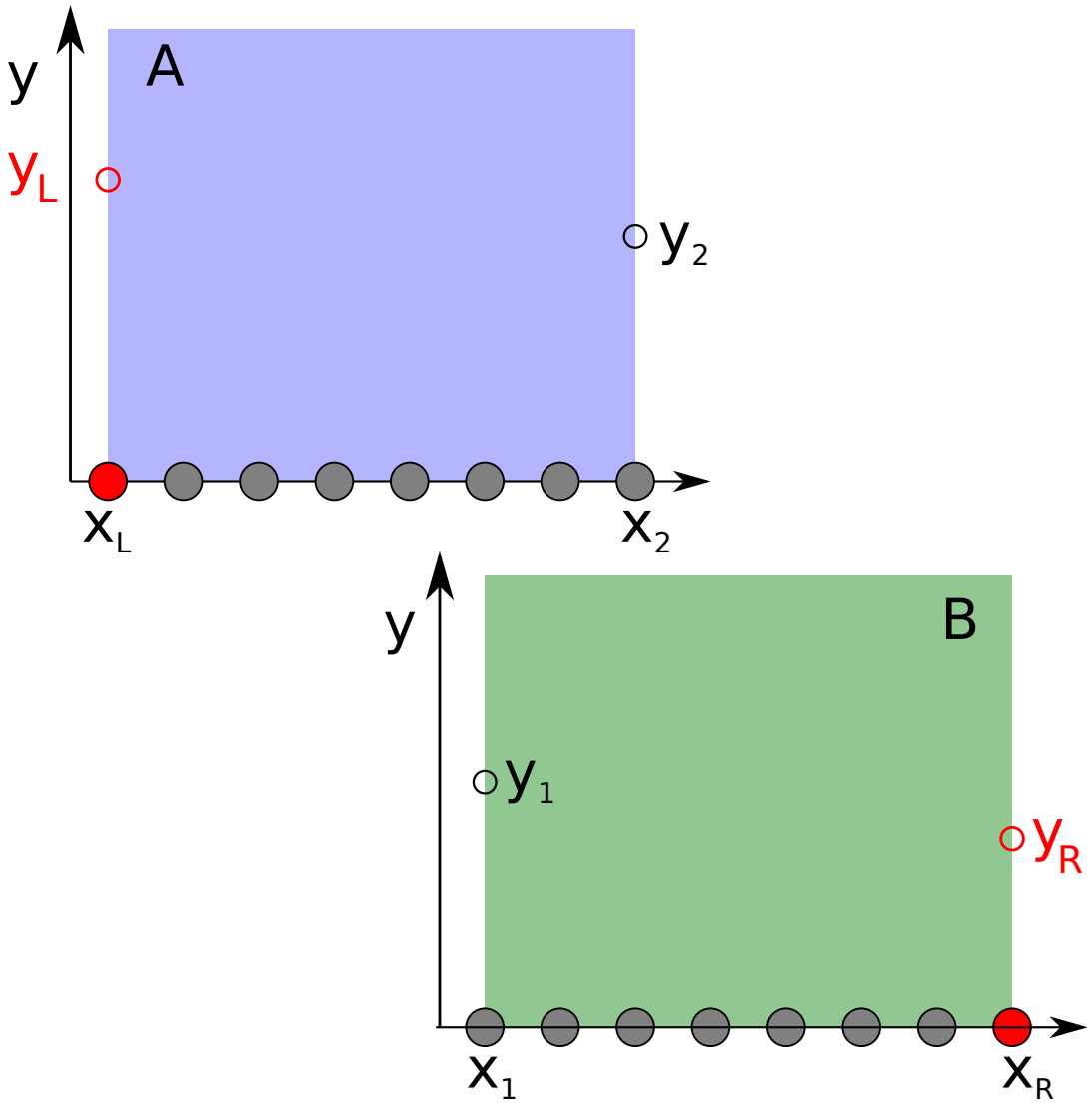
Algorithm Overview: Domain Decomposition & Boundary Maps

- Grid with current state.
- Partition space with overlapping subdomains.
- Treat subdomains independently.
- Define sampling range for each boundary. Sample and solve PDE locally.
- From samples build maps:
  $y_1 = f(y_2, y_L) = a + by_2 + cy_L$
  $y_2 = g(y_1, y_R) = d + ey_1 + fy_R$
  Maps link the subdomains.
- y_L, y_R are known BC.
- Solve, new state: (y_1^*, y_2^*).
- If needed: update range, repeat loop.



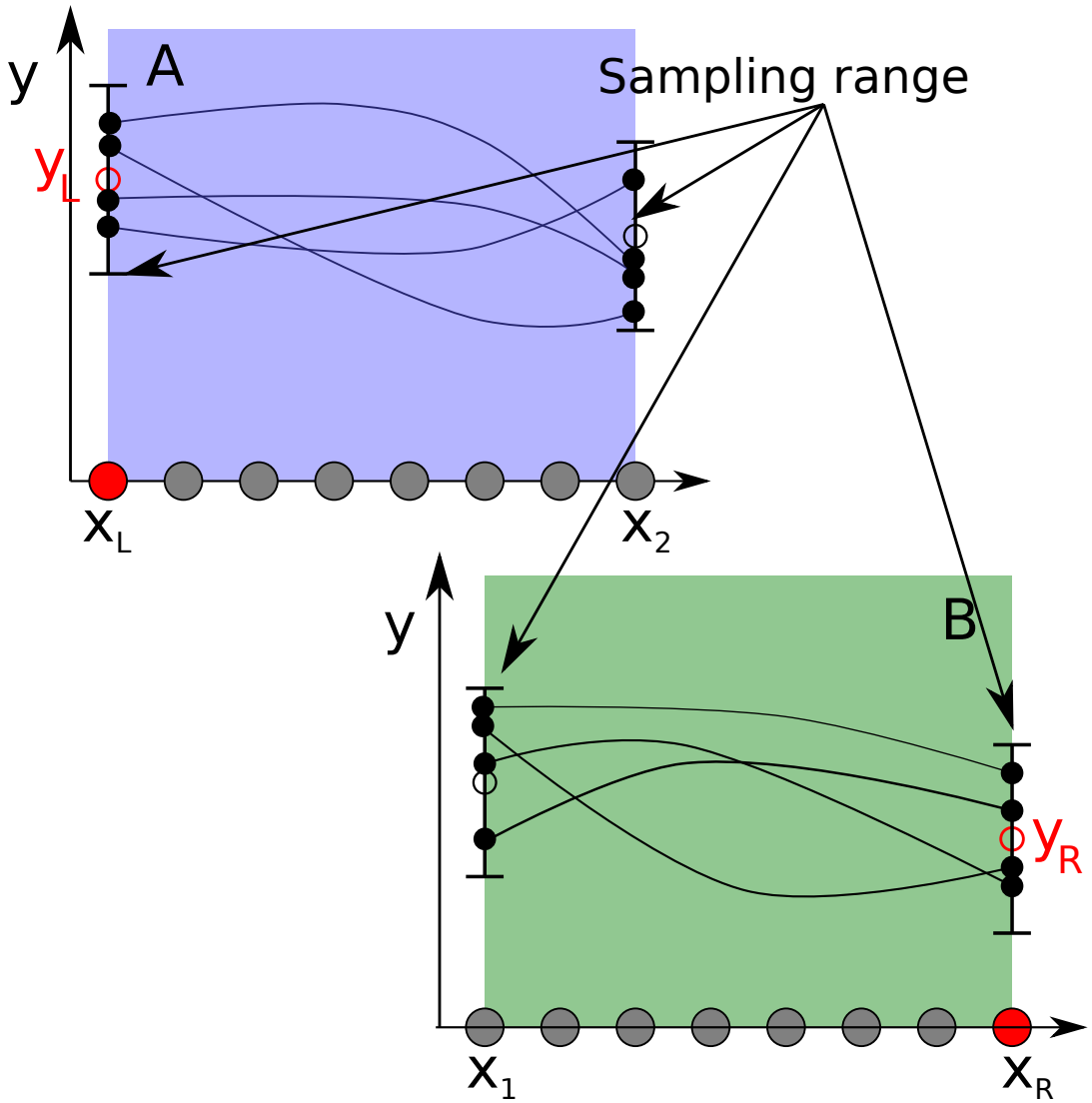
Algorithm Overview: Domain Decomposition & Boundary Maps

- Grid with current state.
- Partition space with overlapping subdomains.
- Treat subdomains independently.
- Define sampling range for each boundary. Sample and solve PDE locally.
- From samples build maps:
  $y_1 = f(y_2, y_L) = a + by_2 + cy_L$
  $y_2 = g(y_1, y_R) = d + ey_1 + fy_R$
  Maps link the subdomains.
- y_L, y_R are known BC.
- Solve, new state: (y_1^*, y_2^*).
- If needed: update range, repeat loop.



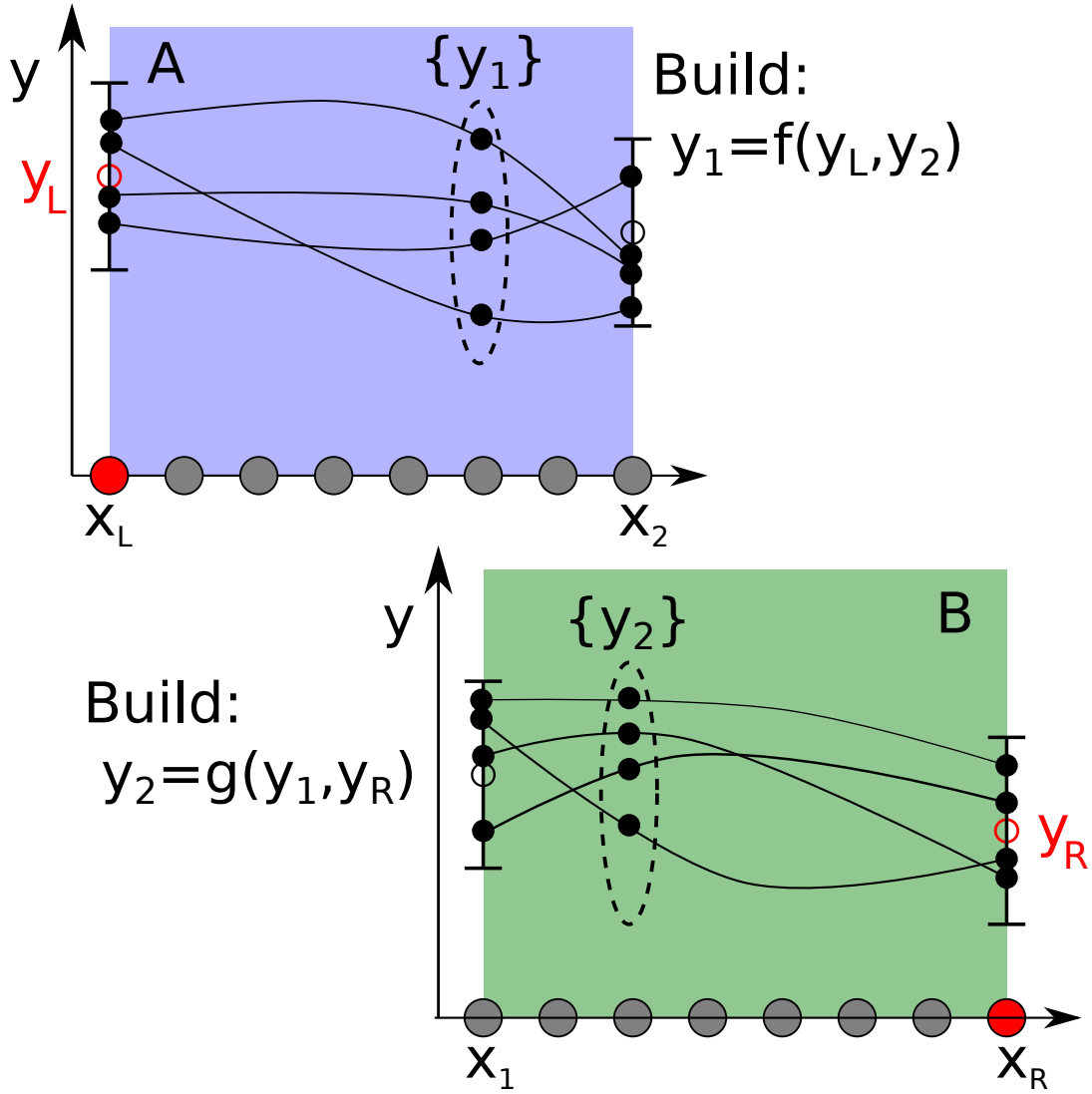
Algorithm Overview: Domain Decomposition & Boundary Maps

- Grid with current state.
- Partition space with overlapping subdomains.
- Treat subdomains independently.
- Define sampling range for each boundary. Sample and solve PDE locally.
- From samples build maps:
  $y_1 = f(y_2, y_L) = a + by_2 + cy_L$
  $y_2 = g(y_1, y_R) = d + ey_1 + fy_R$
  Maps link the subdomains.
- y_L, y_R are known BC.
- Solve, new state: (y_1^*, y_2^*).
- If needed: update range, repeat loop.



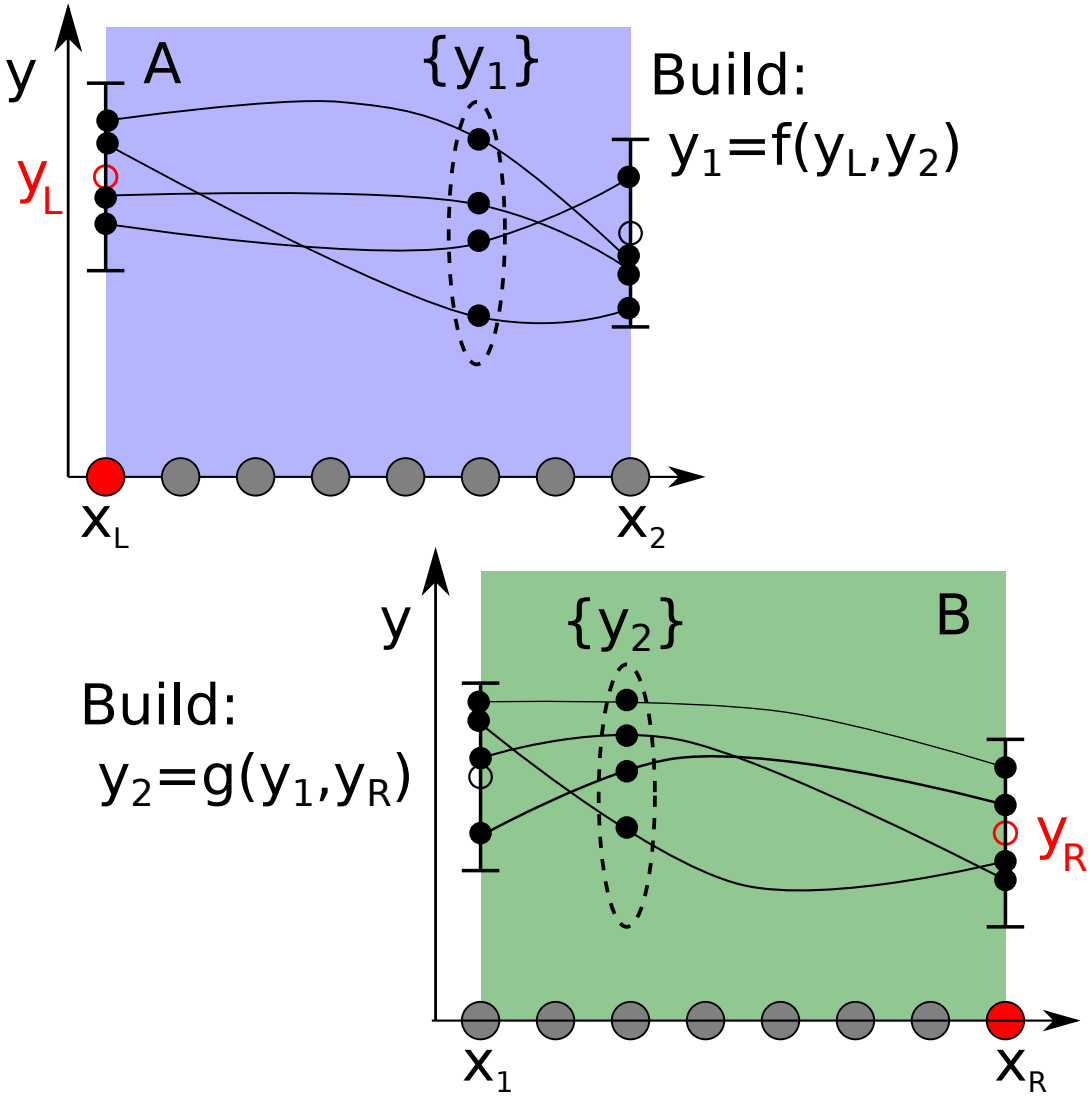
Algorithm Overview: Domain Decomposition & Boundary Maps

- Grid with current state.
- Partition space with overlapping subdomains.
- Treat subdomains independently.
- Define sampling range for each boundary. Sample and solve PDE locally.
- From samples build maps:
  $y_1 = f(y_2, y_L) = a + by_2 + cy_L$
  $y_2 = g(y_1, y_R) = d + ey_1 + fy_R$
  Maps link the subdomains.
- y_L, y_R are known BC.
- Solve, new state: (y_1^*, y_2^*).
- If needed: update range, repeat loop.



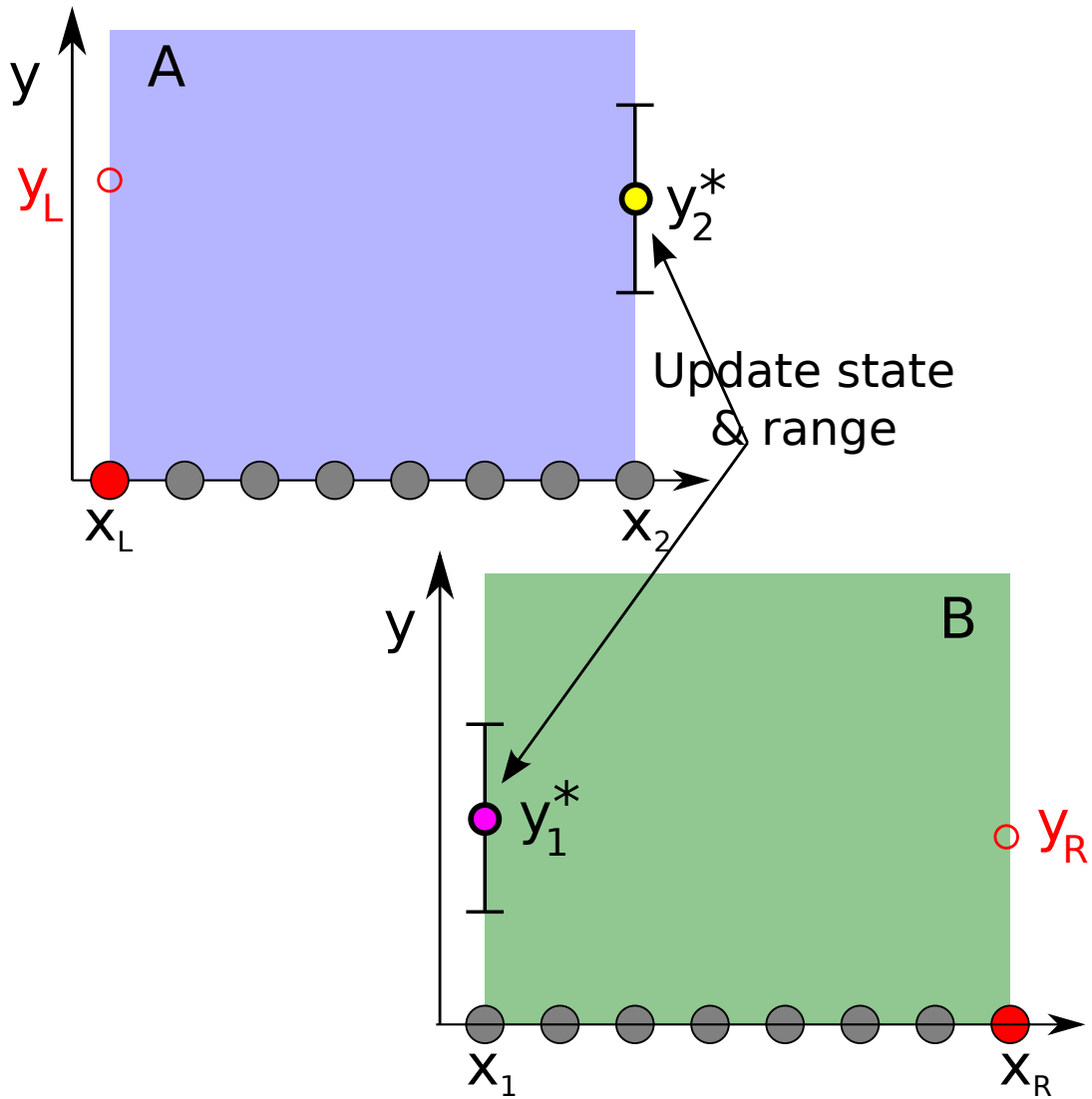
Algorithm Overview: Domain Decomposition & Boundary Maps

- Grid with current state.
- Partition space with overlapping subdomains.
- Treat subdomains independently.
- Define sampling range for each boundary. Sample and solve PDE locally.
- From samples build maps:
  $y_1 = f(y_2, y_L) = a + by_2 + cy_L$
  $y_2 = g(y_1, y_R) = d + ey_1 + fy_R$
  Maps link the subdomains.
- y_L, y_R are known BC.
- Solve, new state: (y_1^*, y_2^*).
- If needed: update range, repeat loop.

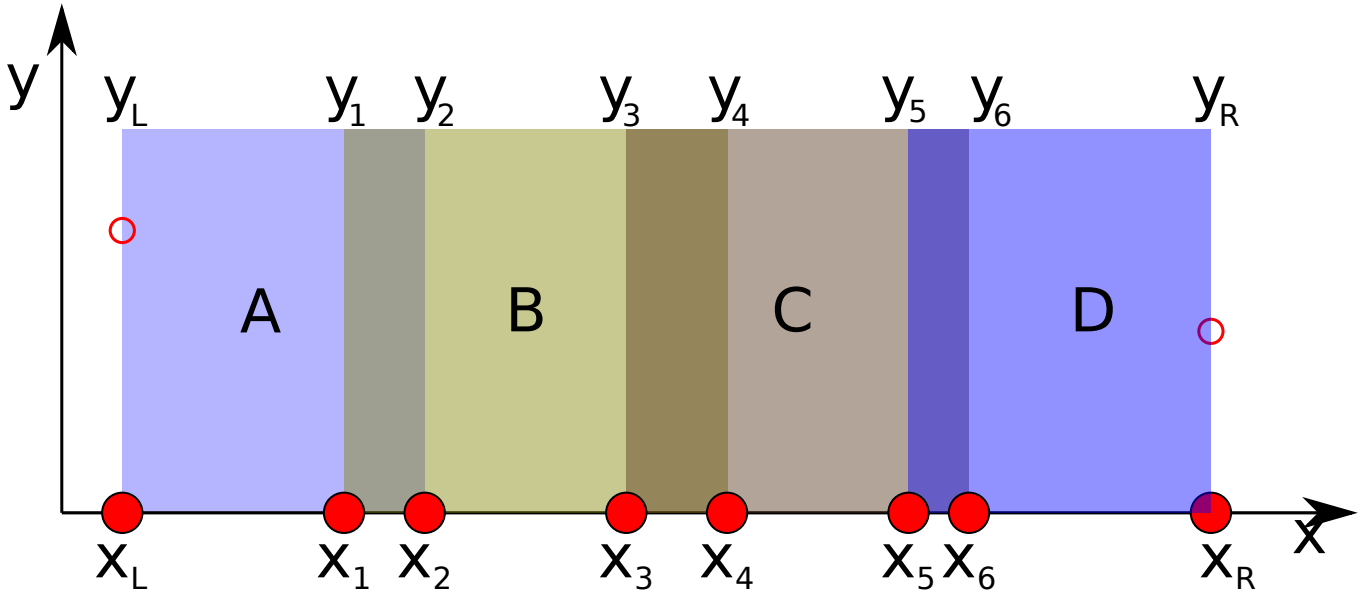


Algorithm Overview: Domain Decomposition & Boundary Maps

- Grid with current state.
- Partition space with overlapping subdomains.
- Treat subdomains independently.
- Define sampling range for each boundary. Sample and solve PDE locally.
- From samples build maps:
  $y_1 = f(y_2, y_L) = a + by_2 + cy_L$
  $y_2 = g(y_1, y_R) = d + ey_1 + fy_R$
  Maps link the subdomains.
- y_L, y_R are known BC.
- Solve, new state: (y_1^*, y_2^*).
- If needed: update range, repeat loop.



Extension

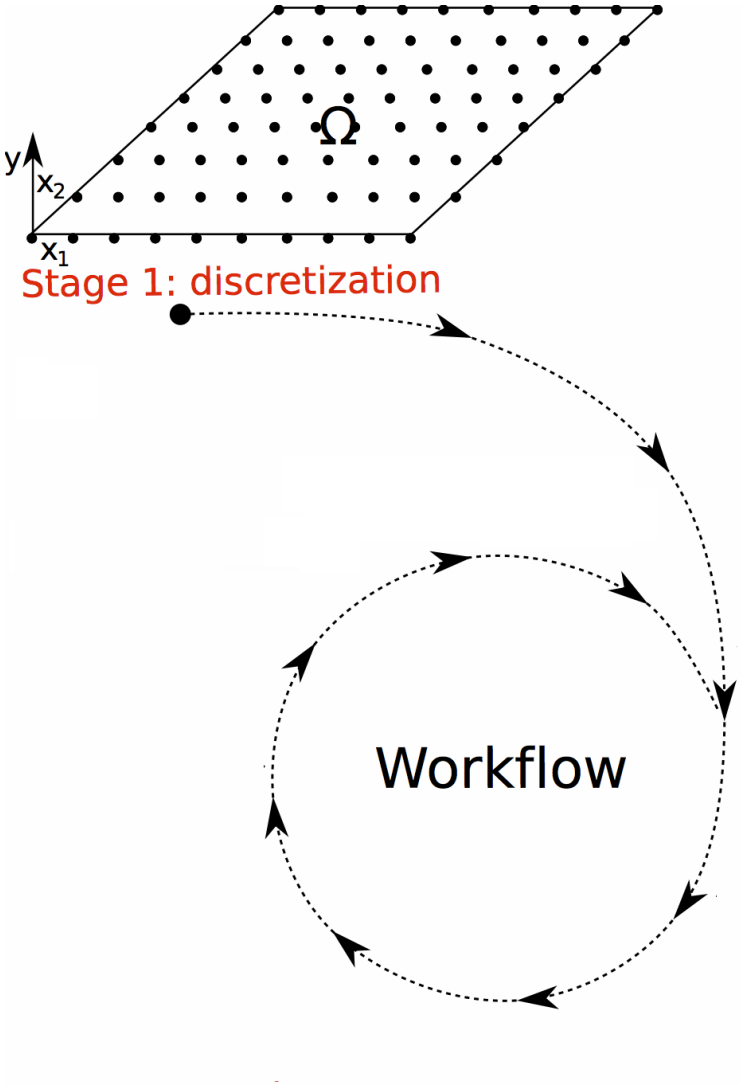


$$y_1 = f_1(y_L, y_2) \qquad y_2 = f_2(y_1, y_4)$$

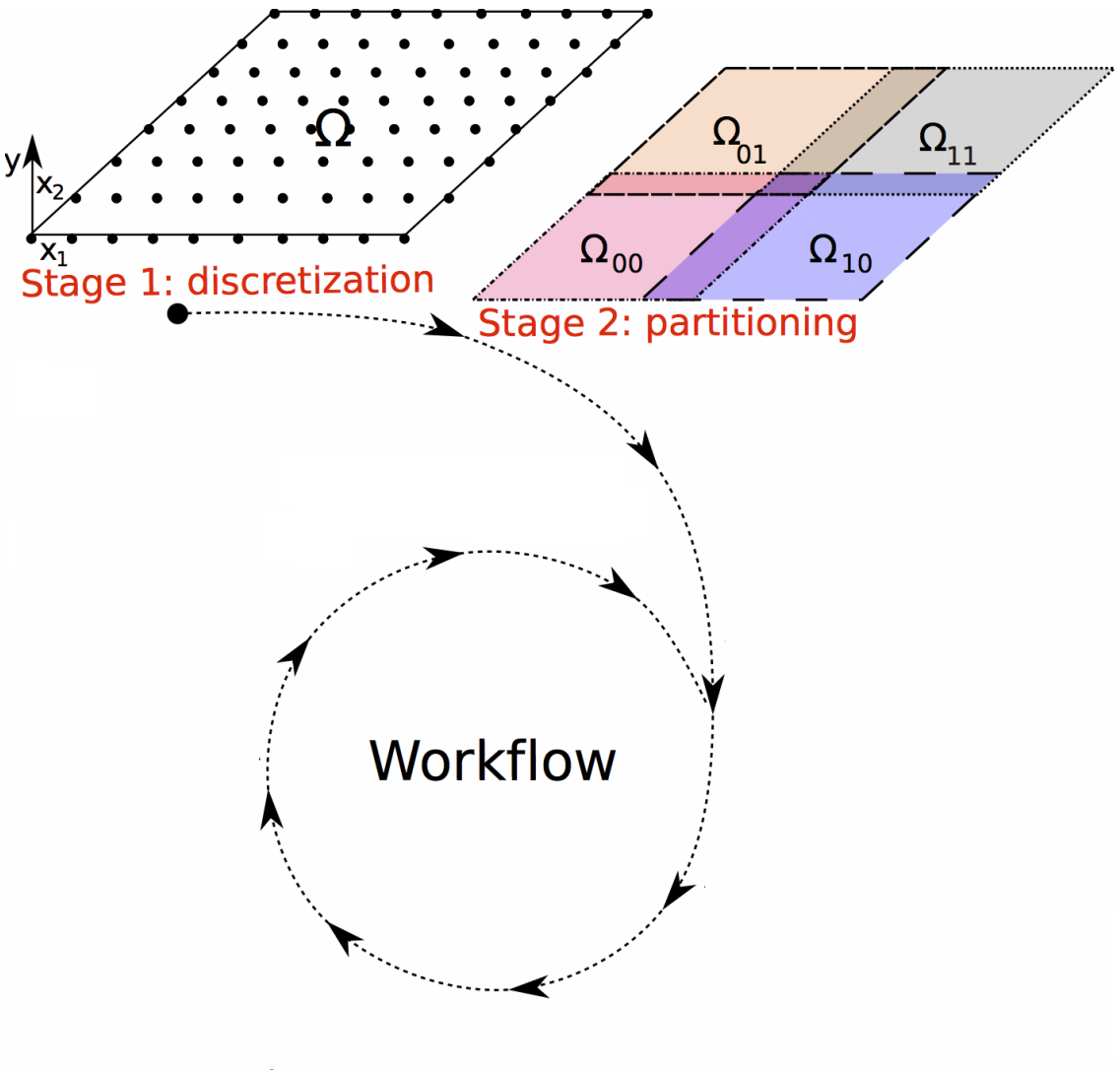
$$y_3 = f_3(y_1, y_4) \qquad y_4 = f_4(y_3, y_6)$$

$$y_5 = f_5(y_3, y_6) \qquad y_6 = f_6(y_5, y_R)$$

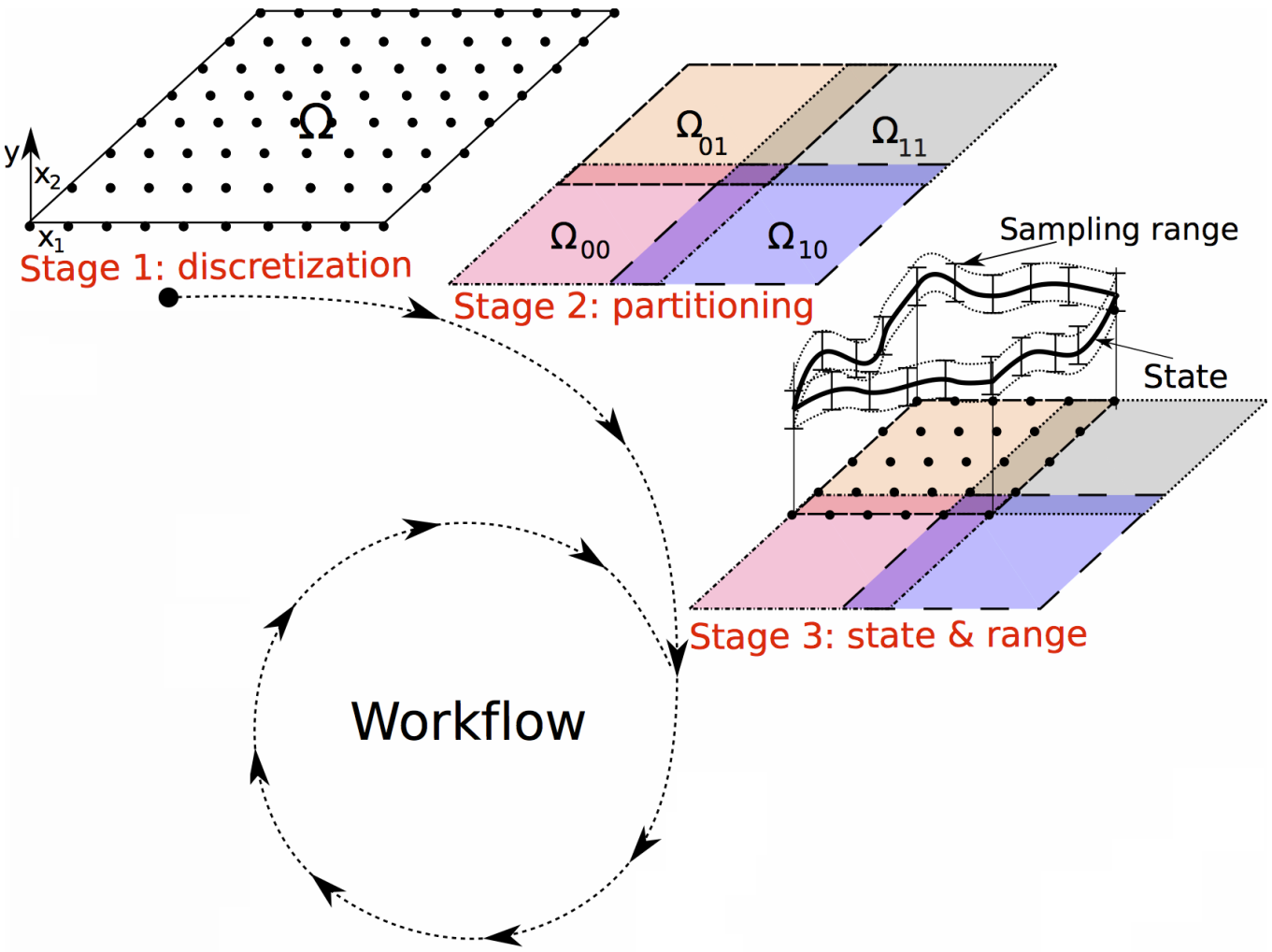
Extension



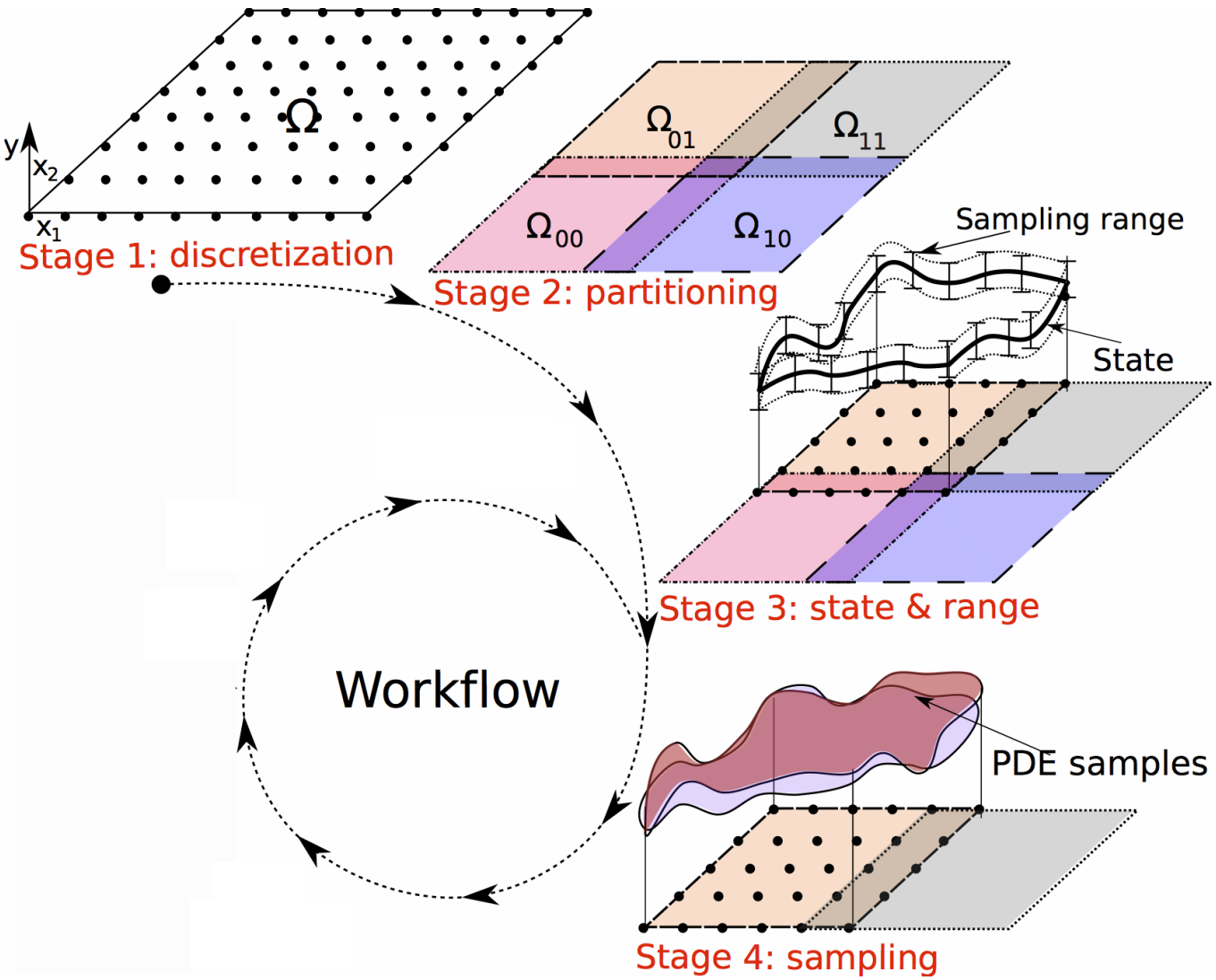
Extension


Ω
y
x2
x1
Stage 1: discretization
Ω01
Ω11
Ω00
Ω10
Stage 2: partitioning
Workflow

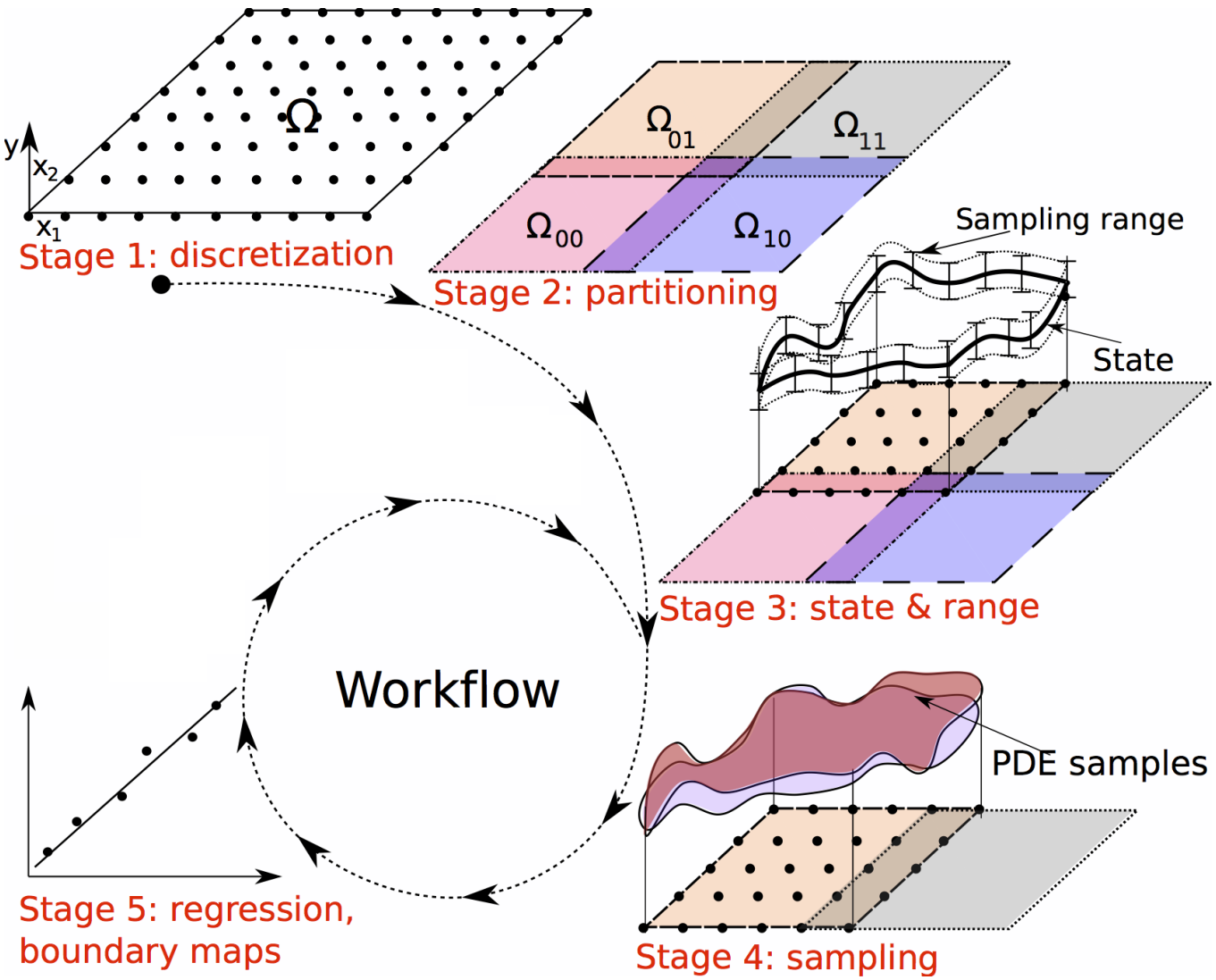
Extension


Ω
y
x2
x1
Stage 1: discretization
Ω01
Ω11
Ω00
Ω10
Stage 2: partitioning
Sampling range
State
Stage 3: state & range
Workflow

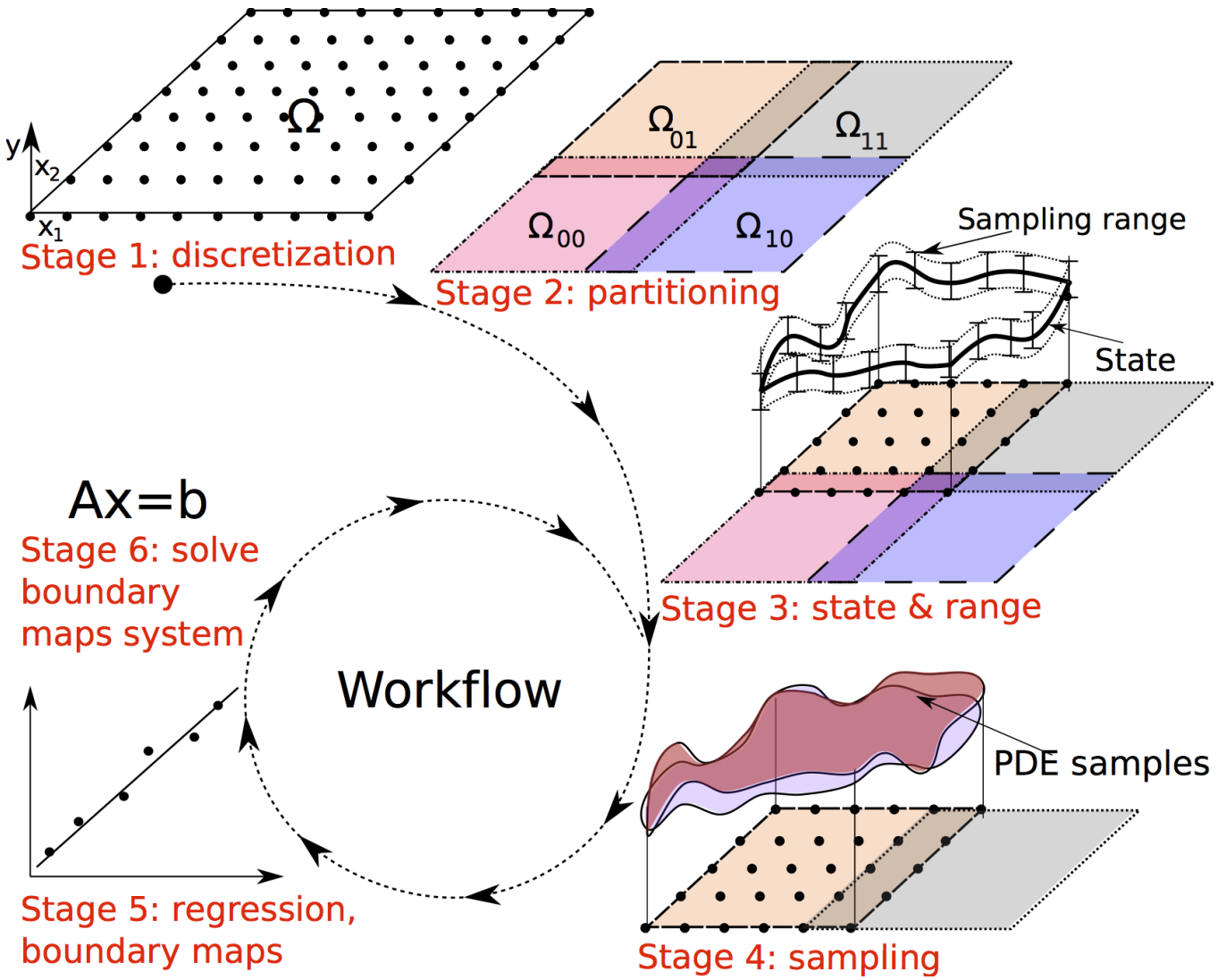
Extension


Ω
y
x2
x1
Stage 1: discretization
Ω01
Ω11
Ω00
Ω10
Stage 2: partitioning
Sampling range
State
Stage 3: state & range
Workflow
PDE samples
Stage 4: sampling

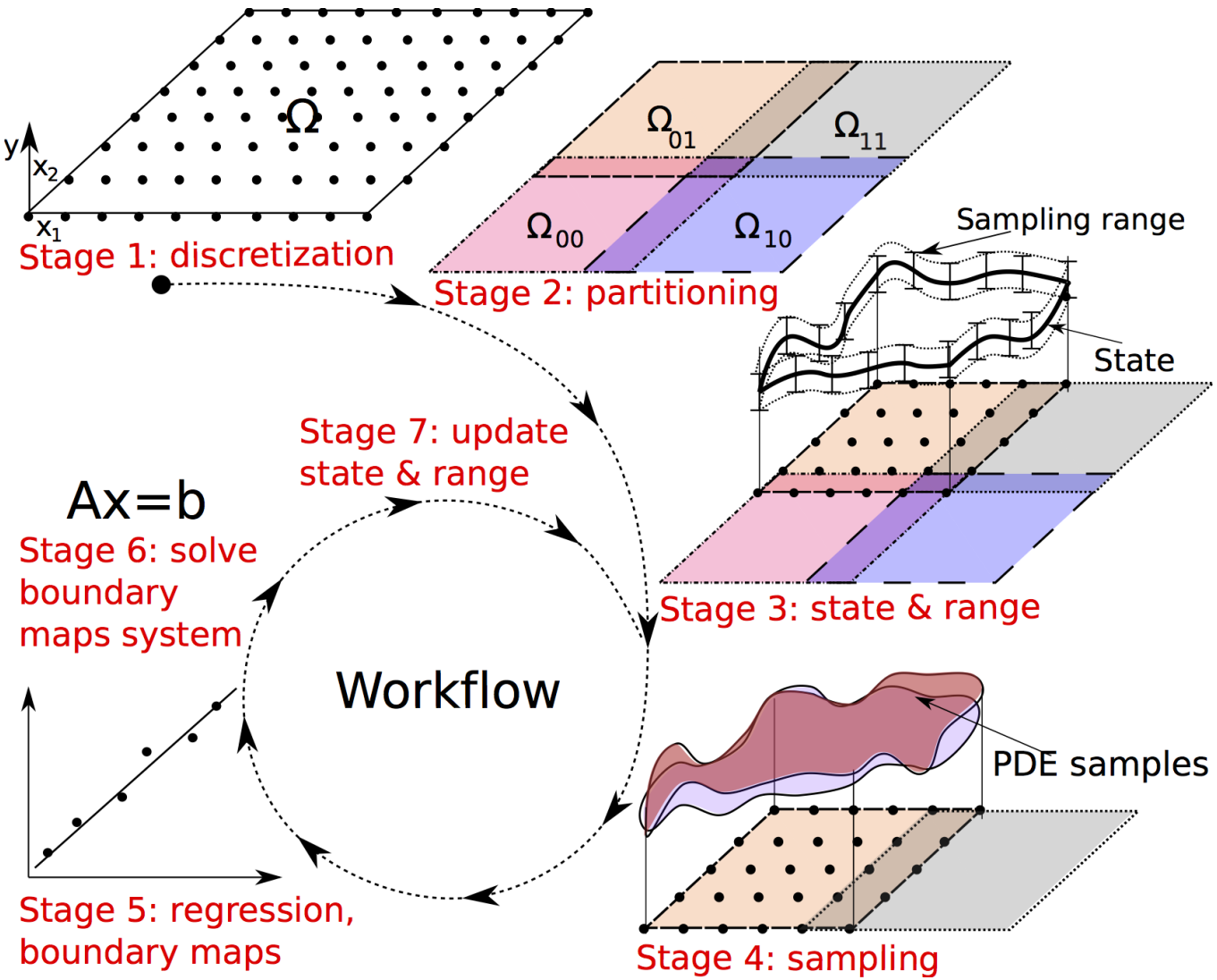
Extension


Ω
y
x2
x1
Stage 1: discretization
Ω01
Ω11
Ω00
Ω10
Stage 2: partitioning
Sampling range
State
Stage 3: state & range
Workflow
PDE samples
Stage 4: sampling
Stage 5: regression,
boundary maps

Extension


Ω
y
x2
x1
Stage 1: discretization
Ω01
Ω11
Ω00
Ω10
Stage 2: partitioning
Sampling range
State
Stage 3: state & range
Ax=b
Stage 6: solve boundary maps system
Workflow
PDE samples
Stage 5: regression, boundary maps
Stage 4: sampling

Extension

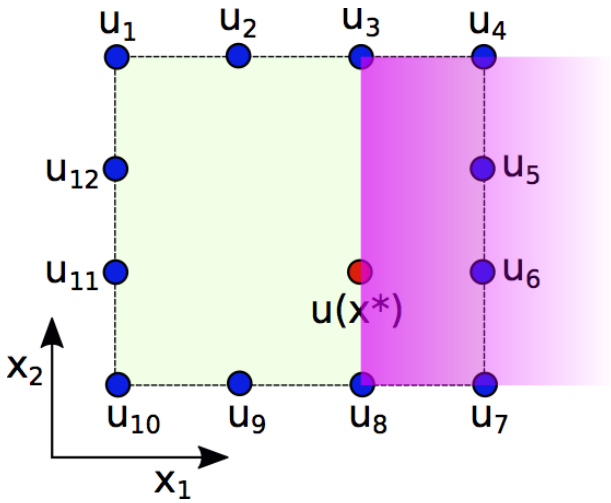

Ω
y
x2
x1
Stage 1: discretization
Ω01
Ω11
Ω00
Ω10
Stage 2: partitioning
Sampling range
State
Stage 3: state & range
Stage 7: update
state & range
Ax=b
Stage 6: solve
boundary
maps system
Workflow
PDE samples
Stage 5: regression,
boundary maps
Stage 4: sampling

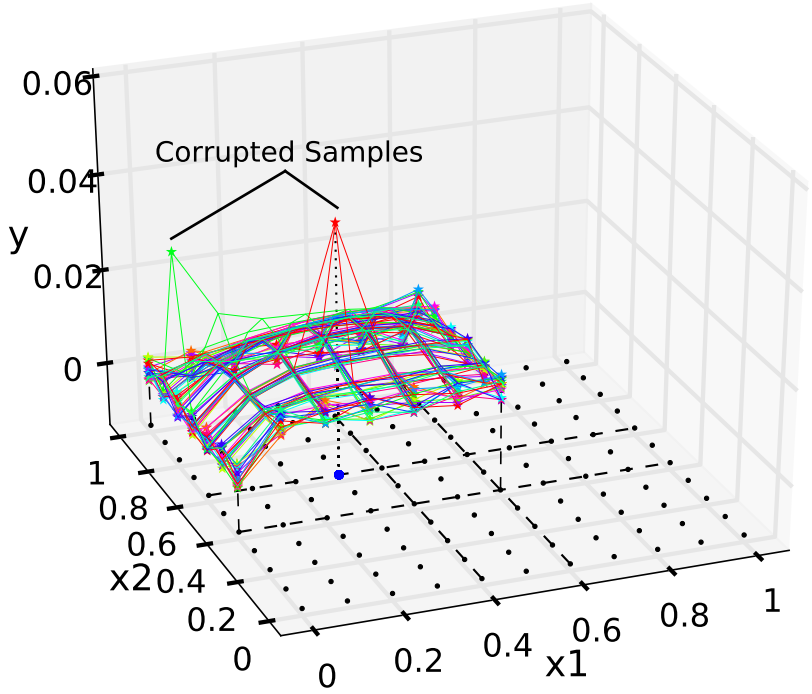
Robust Regression: Resilience to SDCs

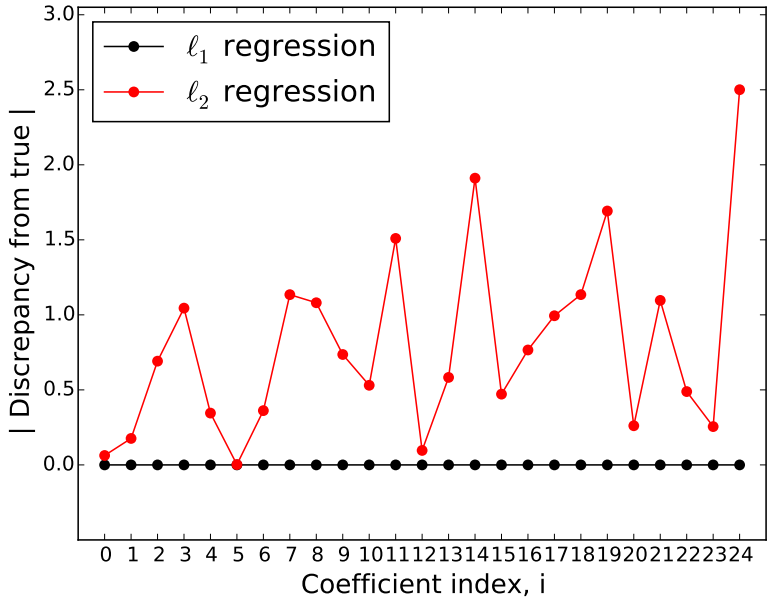
- High-dim boundary maps:

$$u(\boldsymbol{x}^*) = a + bu_1 + cu_2 + \dots$$

- Generate samples, suppose some are corrupted, run regression.
- ℓ_1 noise model is robust against presence of corrupted data.

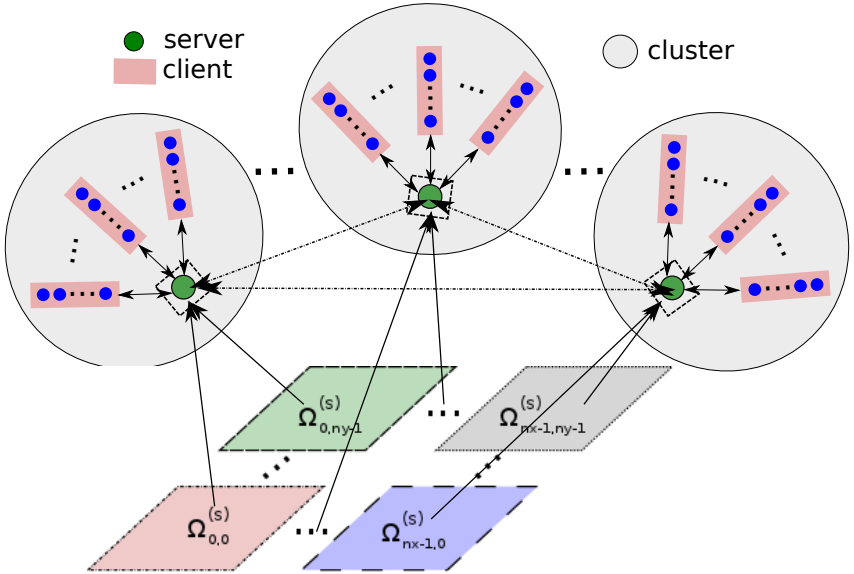






Implementation

Server/Client-based Implementation

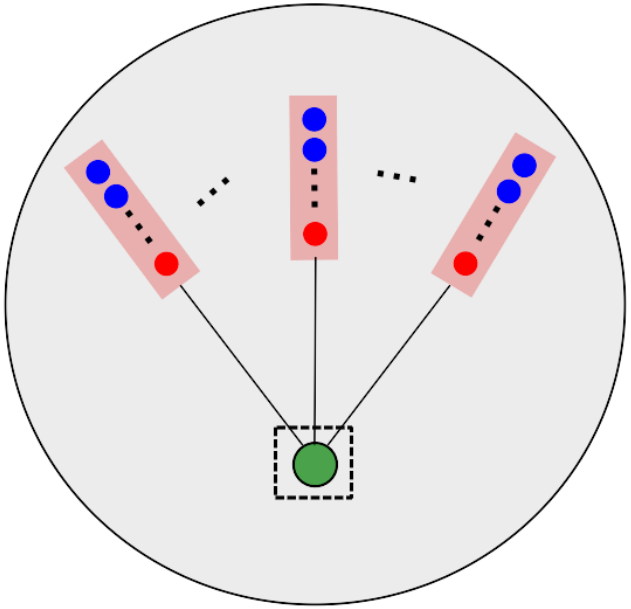


- Cluster: 1 server + *n* clients.
- Servers:
  - Communicate between each other.
  - Safe data/state storage (sandboxed).
- Clients:
  - Independent from one another.
  - Only serve as computing units.

- Separates state from computation: reduces the overall vulnerability.
- Fault-tolerance supported via ULFM-MPI: resilient to MPI ranks crashing.
- Resilient to clients crashing because even if tasks are lost, state is safe.
- It aligns with the vision of future exascale architectures involving heterogeneous and hierarchical hardware required to meet energy and cost constraints.
- C++ code, two external dependencies: Trilinos and Boost.

ULFM and SCM: Why is this a good combination?

- MPI ranks crashing.
- Server simply continues the execution using only the clients that are alive.
  - ✓ Avoid ULFM collective procedures to rebuild the client/communicators.
- Servers probe the corresponding cluster communicator using `MPI_ANY_SOURCE` to assess whether a new message is arriving from one of the clients.
- Client crash:
  1. tell ULFM we are aware of the failure.
  2. ~~Fix communicators, etc.~~
  3. continue normal execution.



ULFM on Edison (NERSC): not an easy task

- Edison: Cray XC30, 2.57 petaflops, 133,824 cores, 357 TB memory. Cray Aries high-speed interconnect with Dragonfly topology.
- Current ULFM-MPI release (1.1) is based on OpenMPI version 1.7.1, which lacks functioning native support for Cray uGNI.
  - × Openib byte transfer layer (BTL)/Infiniband: limite to 4096 MPI ranks.
  - ✓ TCP byte transfer layer does not have this limitation.
- Necessary to set `btl_tcp_if_include = ipogif0` to ensure that TCP/IP packets were sent through the correct network interface. `self` and `sm` BTLs are used for in-node communications.
- This version of OpenMPI has some issues with newer versions of gcc (mostly with string handling), so we resorted to gcc/4.9.3.
- Unloaded the darshan module since that can cause an overhead for large scale jobs.
- With these settings, we were able to have good performance.

Results

Injecting SDC and MPI Ranks Failures

Selective reliability

Inject/perturb applications at target points and evaluate how it behaves.
Parts of the algorithm are assumed to be handled more reliably than others.
M.Hoemmen,M.Heroux,2012

Silent Data Corruptions (SDC)

- Injected during the sampling stage.
- Injection at random as % of tasks.
- Corrupt all boundary conditions data of a task.
- How: double -> binary -> random bit-flip -> convert to double.

MPI Ranks Failures

- Injected during the sampling stage.
- Injection occurs at random.
- Actually terminate (`SIGKILL`) the processes owing those ranks.

Injecting SDC and MPI Ranks Failures

Selective reliability

Inject/perturb applications at target points and evaluate how it behaves.
Parts of the algorithm are assumed to be handled more reliably than others.
M.Hoemmen,M.Heroux,2012

Silent Data Corruptions (SDC)

- Injected during the sampling stage.
- Injection at random as % of tasks.
- Corrupt all boundary conditions data of a task.
- How: double -> binary -> random bit-flip -> convert to double.

MPI Ranks Failures

- Injected during the sampling stage.
- Injection occurs at random.
- Actually terminate (`SIGKILL`) the processes owing those ranks.

Injecting SDC and MPI Ranks Failures

Selective reliability

Inject/perturb applications at target points and evaluate how it behaves.
Parts of the algorithm are assumed to be handled more reliably than others.
M.Hoemmen,M.Heroux,2012

Silent Data Corruptions (SDC)

- Injected during the sampling stage.
- Injection at random as % of tasks.
- Corrupt all boundary conditions data of a task.
- How: double -> binary -> random bit-flip -> convert to double.

MPI Ranks Failures

- Injected during the sampling stage.
- Injection occurs at random.
- Actually terminate (`SIGKILL`) the processes owing those ranks.

Details

Resilience Condition:

- Out of the samples used in the regression, the number of uncorrupted samples has to be greater than the minimum set needed to have a well-posed regression problem.
- Ensured via oversampling: $\rho > 1$, such that $N = \rho N^s_{nom}$.

  N^s_{nom}: number of samples for the fault-free scenario.

Silent Data Corruptions (SDC)

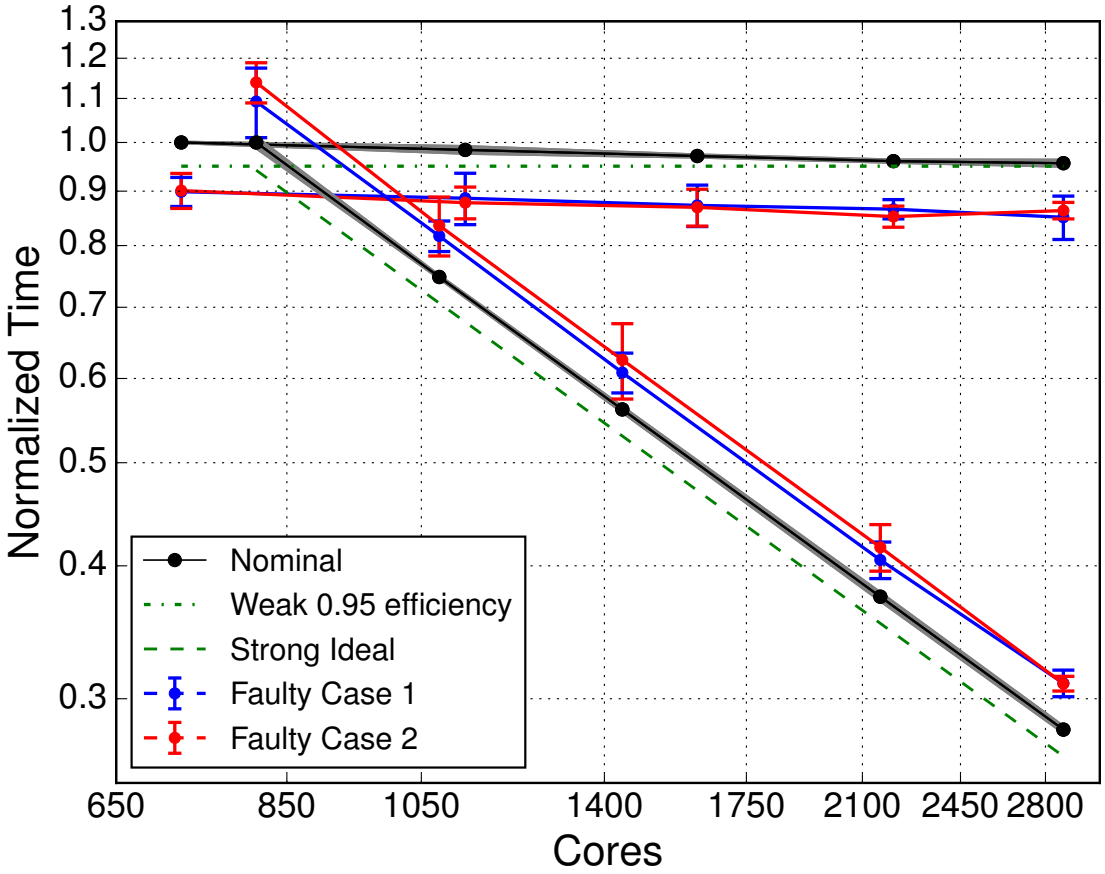
- The ℓ_1-model enables resilience to corrupted data.

MPI Ranks Failures

- Server continues the execution using only the clients that are alive.
- No need for ULFM collectives to rebuild broken communicators.

Kahuna Scaling: Uncertainty due to SDCs and client failures

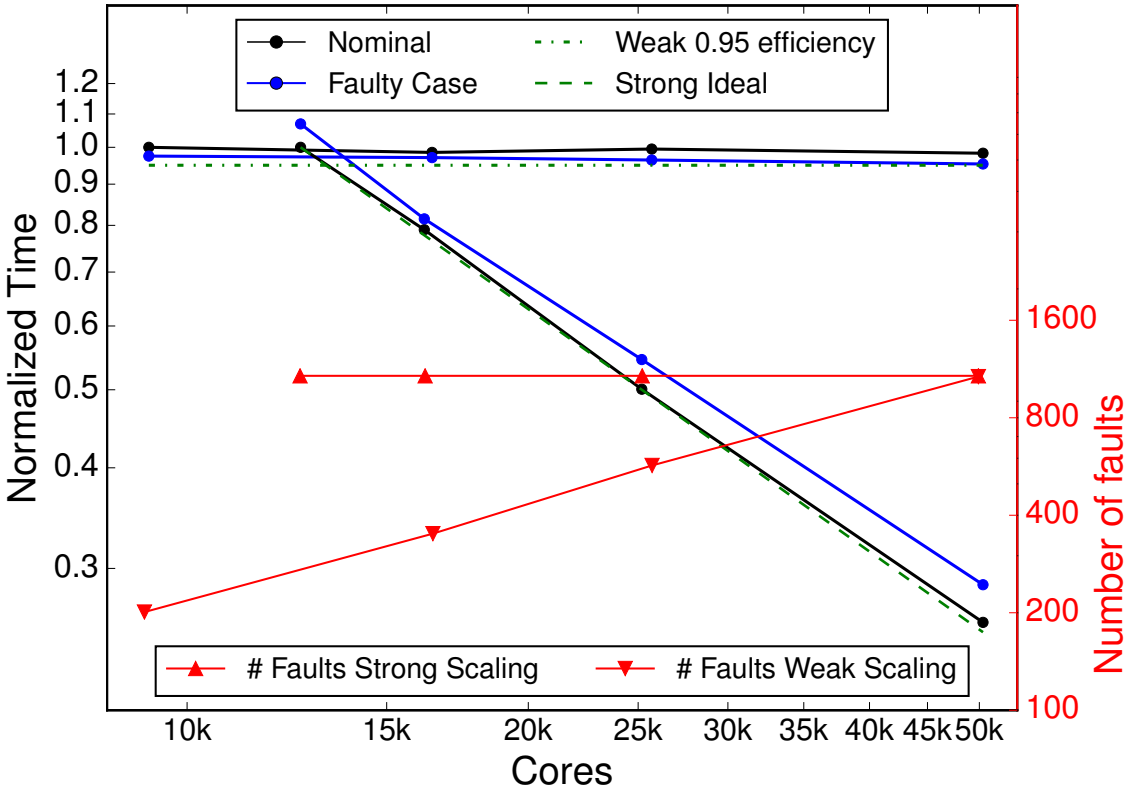
- Kahuna: total 3080 cores.
- Blue: 1% clients failing, SDCs=0.1% of total tasks.
- Red: 2% clients failing, SDCs=0.1% of total tasks.
- Oversampling $\rho = 1.05$.
- 4 samples: mean, errors $\pm 3\sigma$.
- Times normalized by smallest nominal case: highlights scaling and overhead wrt nominal.



- Black for NOMINAL: excellent scalability and minimal variability.
- Faulty runs show larger variability due to random loss of resources, and additional regression overhead to overcome SDCs.
- Red has larger overhead because number of failed clients doubles.

Edison Scaling

- Edison, elliptic PDE.
- Of total faults: SDCs=97%, ranks failures=3.0%.
- Constant machine fault rate.
- Oversampling of $\rho = 1.05$.
- Times normalized by smallest nominal case: highlights scaling and overhead wrt nominal.



- Excellent scalability with and without faults:

  within 95% for weak scaling, withn 90% for strong.

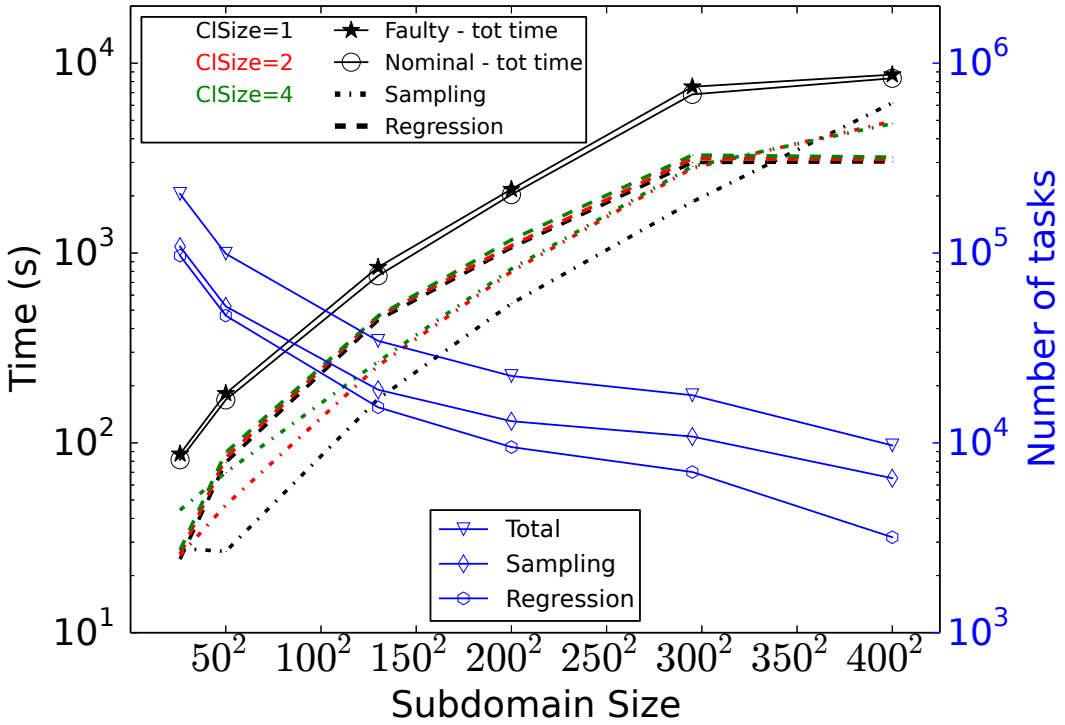
- Overhead wrt nominal case:

  for weak it shows as downward shift, for strong as upward shift.

Subdomain Size

Effect of subdomain size on execution time, and its interplay with computational workload, communication costs and size of the clients.

- Use in-house cluster.
- SCM with fixed 36 client ranks:
  - 36 clients size 1
  - 18 clients size 2
  - 9 clients size 4
- 5% of the clients are terminated; corrupt 0.1% samplings tasks.
- Multiple samples for each case and extract mean values.
- Overhead due to faults is small.
- Best performance is obtained for small subdomains.



Power Consumption

- Resilience and energy consumption are tightly linked: voltage decrease is linked to higher faults rates. [a]
- Decreasing the energy consumption is possible via variable-voltage CPUs, which can reduce power consumption quadratically at the expense of linearly reduced speed . [b]
- Another possibility is for given frequency, to decrease voltage only up to the a certain threshold.

[a] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems", IEEE-ACM International Conference, 2004

[b] D. Zhu, R. Melhem, D. Mosse, and E. Elnozahy, "Analysis of an energy efficient optimistic tmr scheme", ICPADS 2004

Power Consumption

- How to exploit the resilience of the application and the small overhead for energy purposes?
- Idea: lower the energy consumption during the *sampling stage* by means of voltage scaling.
- Compare three scenarios:
  (A) machine running at full operational capacity/speed
  (B) voltage/frequency scaling on clients during sampling
  (C) voltage scaling on clients during sampling
- Same problem, same SC configuration, same machine.
- The servers always run at full capacity to keep the state safe.
- This framework can be enabled because of the SCM, which allows us to separate state from computation.

Power Consumption

Power consumption and energy over $T = t_2 - t_1$ [a]:

$$P = \hat{P} + CV^2 f \qquad\qquad E = (\hat{P} + CV^2 f)T.$$

$\hat{P}$ = frequency independent active power

C is the switch capacitance

V is the voltage, and f is the frequency.

no sleep power: system always on.

[a] D. Zhu, R. Melhem, and D. Mosse, IEEE-ACM, 2004

Power Consumption

Full operational mode: (A)

- V_A, f_A
- $t_A^s =$ time for one task.
- Energy for N_A samples: $E_A^s = N_A(\hat{P} t_A^s + \mathcal{C} V_A^2 f_A t_A^s)$

Reduced voltage/frequency: (B)

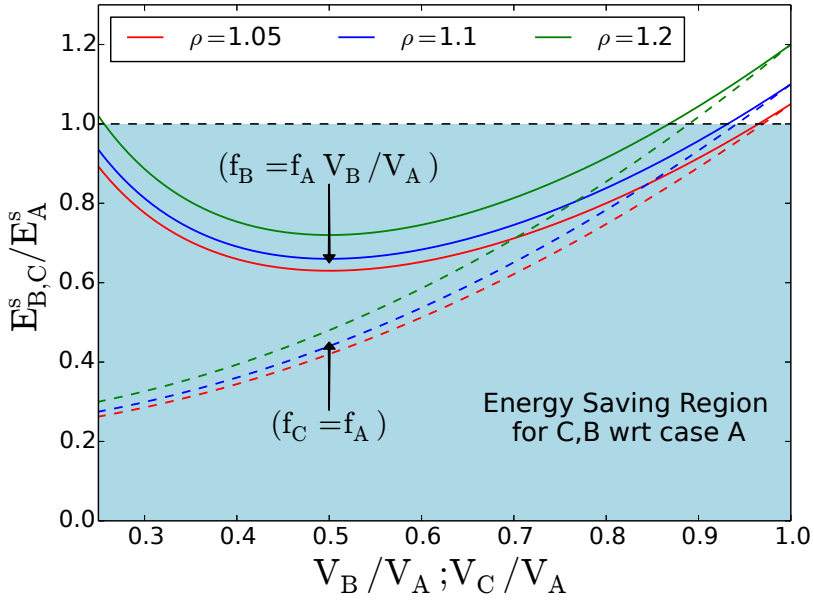
- $V_B < V_A, f_B < f_A$ such that $V_B / V_A = f_B / f_A$.
- $t_B^s = t_A^s \frac{f_A}{f_B}$.
- Energy for $N_B = \rho N_A$ samples: $E_B^s = \rho N_A(\hat{P} t_A^s \frac{f_A}{f_B} + \mathcal{C} V_A^2 f_A t_A^s \frac{f_B^2}{f_A^2})$

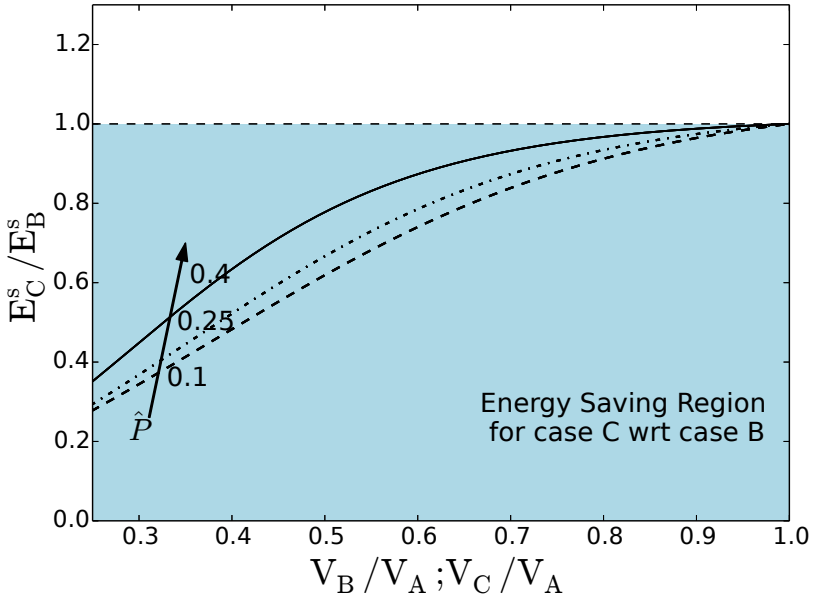
Reduced voltage: (C)

- $V_C = \gamma V_A$, with $\gamma < 1$, $f_C = f_A$.
- $t_C^s = t_A^s$.
- Energy for $N_C = \rho N_A$ samples: $E_C^s = \rho N_A(\hat{P} t_A^s + \mathcal{C} \gamma^2 V_A^2 f_A t_A^s)$.

Power Consumption

- Voltage scaling causes fault rates increase exponentially.
- For *full* application: energy usage due to regression overhead is smaller than the energy gain during sampling.
- Kahuna: 6% regression overhead due to 5% oversampling, 1% client failures, and 0.1% SDCs.
- Case(B), left fig, $\rho = 1.05$:
  20% voltage reduction $->\sim$ 20% potential saving during sampling.
- $->$ 5% regression overhead $->$ net 15% energy gain.





Conclusions and Ongoing Work

- Application is resilient to:
  - Silent Data Corruptions.
  - MPI ranks failing.
- Convergence is achieved in all cases.
- Sampling/decomposition approach provides concurrency/parallelism.
- Scalability is excellent with and without faults.
- Interesting tradeoffs between energy and resilience.
- Ongoing work/outlook:
  - Dimensionality reduction.
  - Extension to other types of PDE.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number 13-016717.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

THANK YOU