Performance oriented prediction based on improved Gaussian process regression

Dinh-Mao Bui

Advisor:
Professor **Sungyoung Lee**

01st November, 2017

Department of Computer Science and Engineering
Kyung Hee University

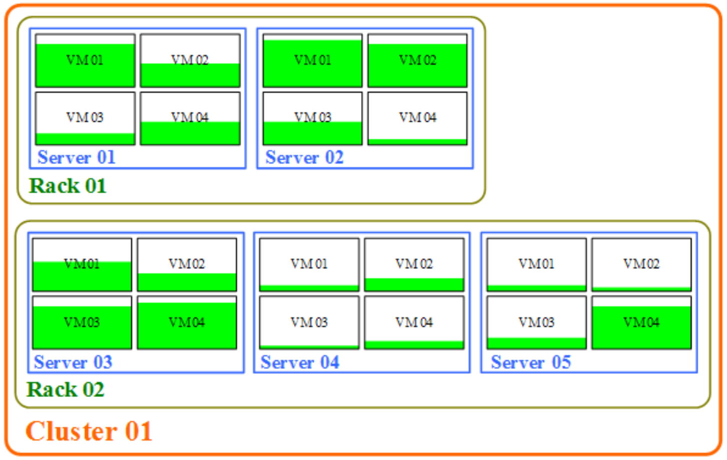
Table of content

Background (1/2)

- To enhance the operation of distributed system, monitoring tool is often used
  - Monitor the state
  - Reconfigure internal components
  - Orchestrate the operation
- Monitoring information is data source for decision making.
  - Ordered in time: **time-series** data.
- However, monitoring system always provides delayed information.
  - Delay makes data obsolete.
  - Affect the precision of decision.

➔ Need for data prediction to make the information more updated.

➔ Better information produces better decision.



A typical distributed system.

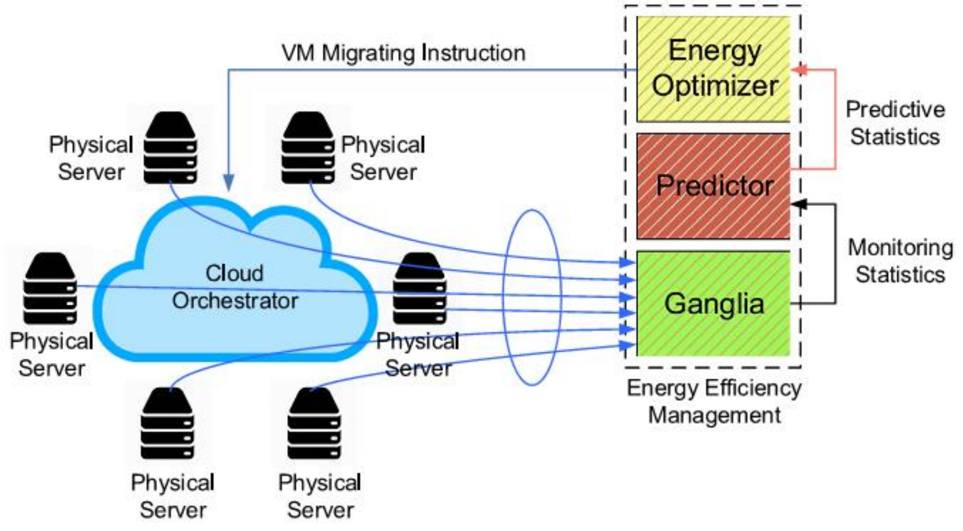
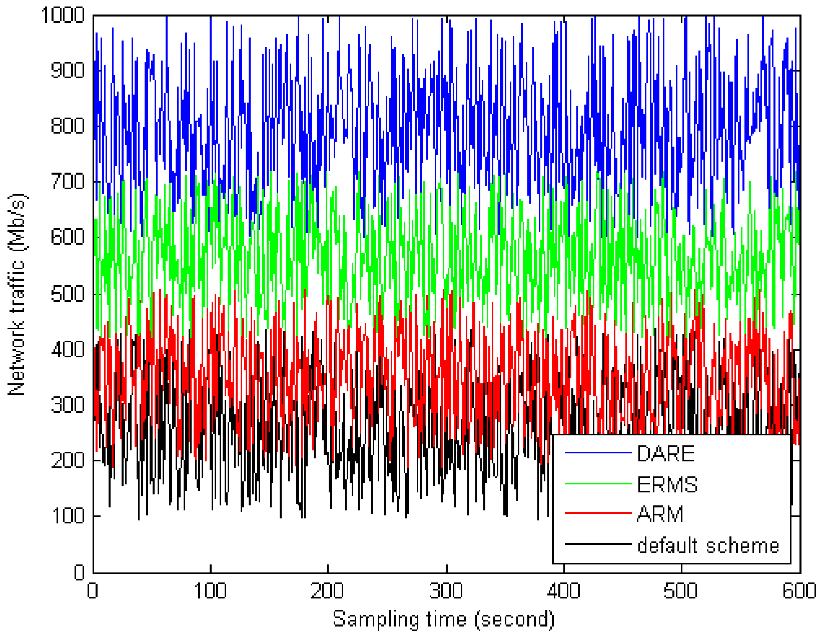


Fig. 2. Architecture of energy efficiency management (E2M) system.



Monitoring data on networking.

Background (2/2)

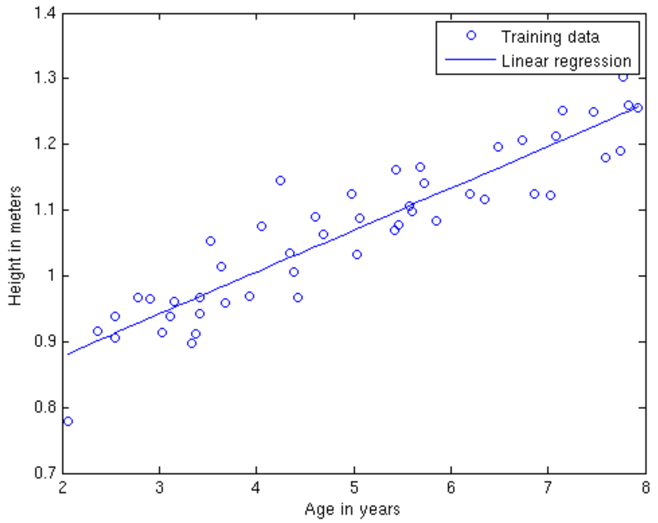
Training points
Predictive value

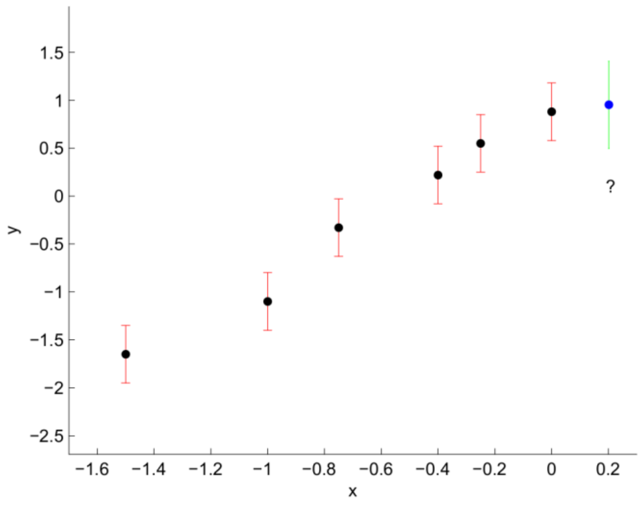
- Definition of prediction: regression technique
  - Statistical technique.
  - Explore the relationships between variables.
  - Use to predict/estimate futuristic values.
  - Type of regression technique is various
    - Linear regression
    - Bayesian regression
    - Ridge regression
    - …
  - How to choose: based on the characteristic of the data
- Definitions: Gaussian process regression
  - Underlying function: *y = f(x)*
  - *f(x)*: linear, quadratic, cubic, non-polynomial…

  → Use model selection to choose the best-fit function.
  - Better solution: Gaussian process regression (GPR).
    - Consists of two phases:
      - **Hyper-parameters learning phase.**
      - **Training phase.**

→ Let data 'speak' clearly about themselves.

→ To be more accurate prediction technique.





Motivation

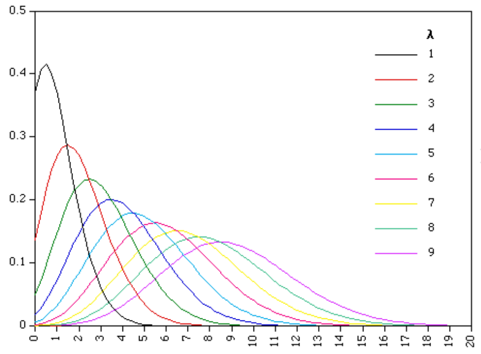
- Characteristic of monitoring data in distributed system: related to the arrival rate of tasks coming to servers.
  - Distribution of tasks follows Poisson distribution

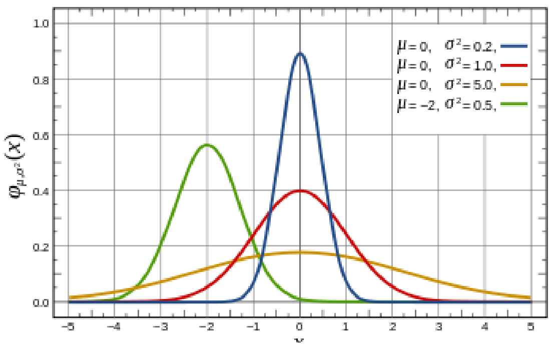
$$f(k;\lambda) = \Pr(X=k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad (1)$$

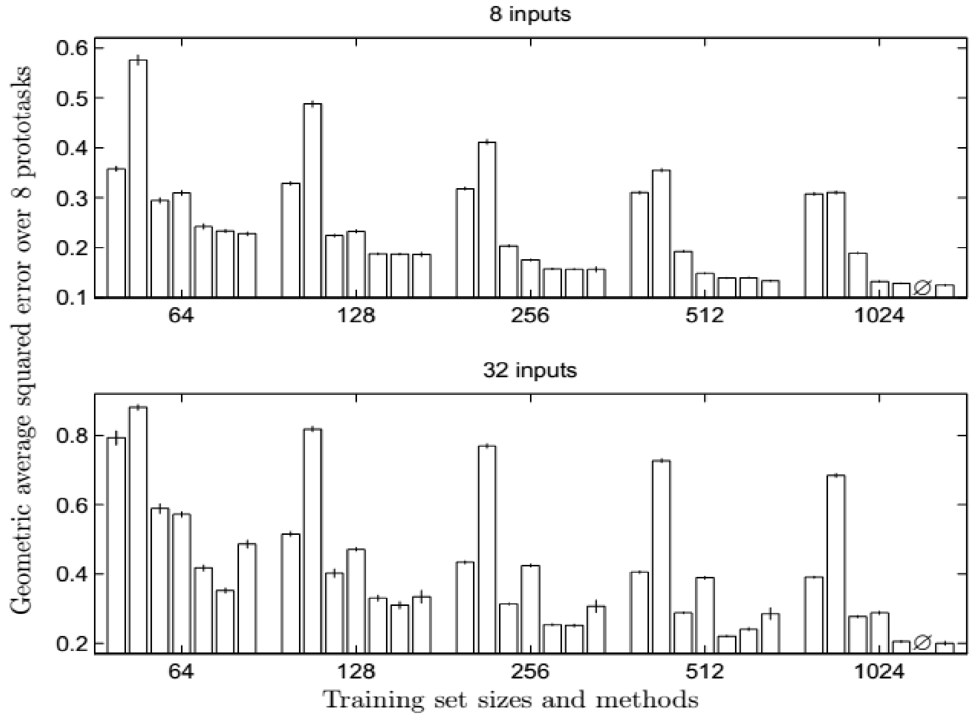
  - According to the relationship between probabilistic distributions[*], when arrival rate (λ) increases, Poisson distribution converges to Gaussian distribution.
  - Gaussian distribution over time establishes Gaussian process.

  ➔ Using GPR to handle the monitoring data is natural.

- Why choose Gaussian process?
  - Accuracy [**]
    - Linear regression
    - K – nearest neighbor (knn)
    - Multivariate adaptive regression splines (MARS)
    - Multilayer Perceptron Ensembles trained with Early Stopping (MLP-ESE)
    - Gaussian process regression - Maximum a posteriori estimation (GPR - MAP)
    - Gaussian process regression – Markov chain Monte Carlo (GPR - MC)
    - Multilayer Perceptron - Markov chain Monte Carlo (MLP-MC)







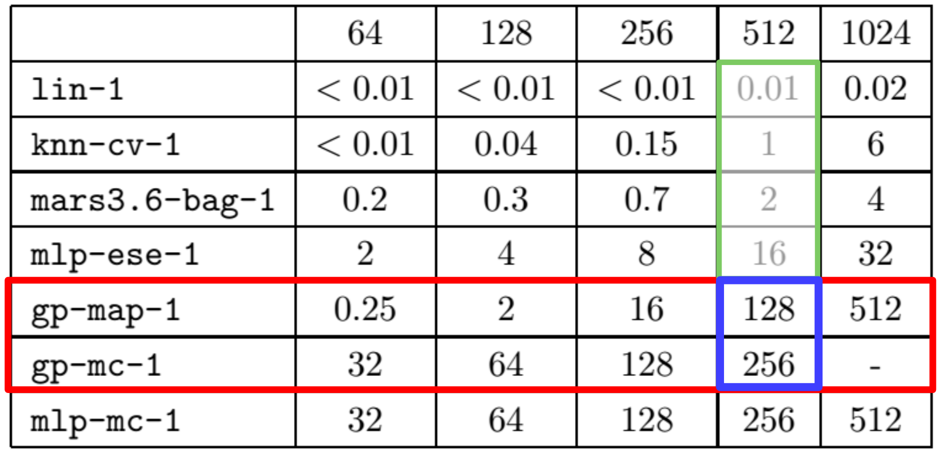
[*] Gallager, R.G., 2013. Stochastic processes: theory for applications. Cambridge University Press.

[**] Experiments from the publication: Rasmussen, C. E. (1996). *Evaluation of Gaussian processes and other methods for non-linear regression* (Doctoral dissertation, University of Toronto).

Problem statement

Training points
Gaussian process regression
Predictive value

- Problem of Gaussian process regression
  - Reaction rate: CPU time evaluated in second for regression.
  - Among the famous prediction techniques, Gaussian process family suffer the critical degradation in processing speed.
- Cause of problem
  - GPR costs O(n^3) for computational complexity when calculating *n* training points of a dataset.
    - Matrix inverse.
    - Log determinant calculation for the covariance matrix.
- Objective: solve the performance issue of GPR.
- Research values
  - Innovate the development of many applications based on GPR.
  - Improve the current GP-family approaches in data communications, networking, signal processing…



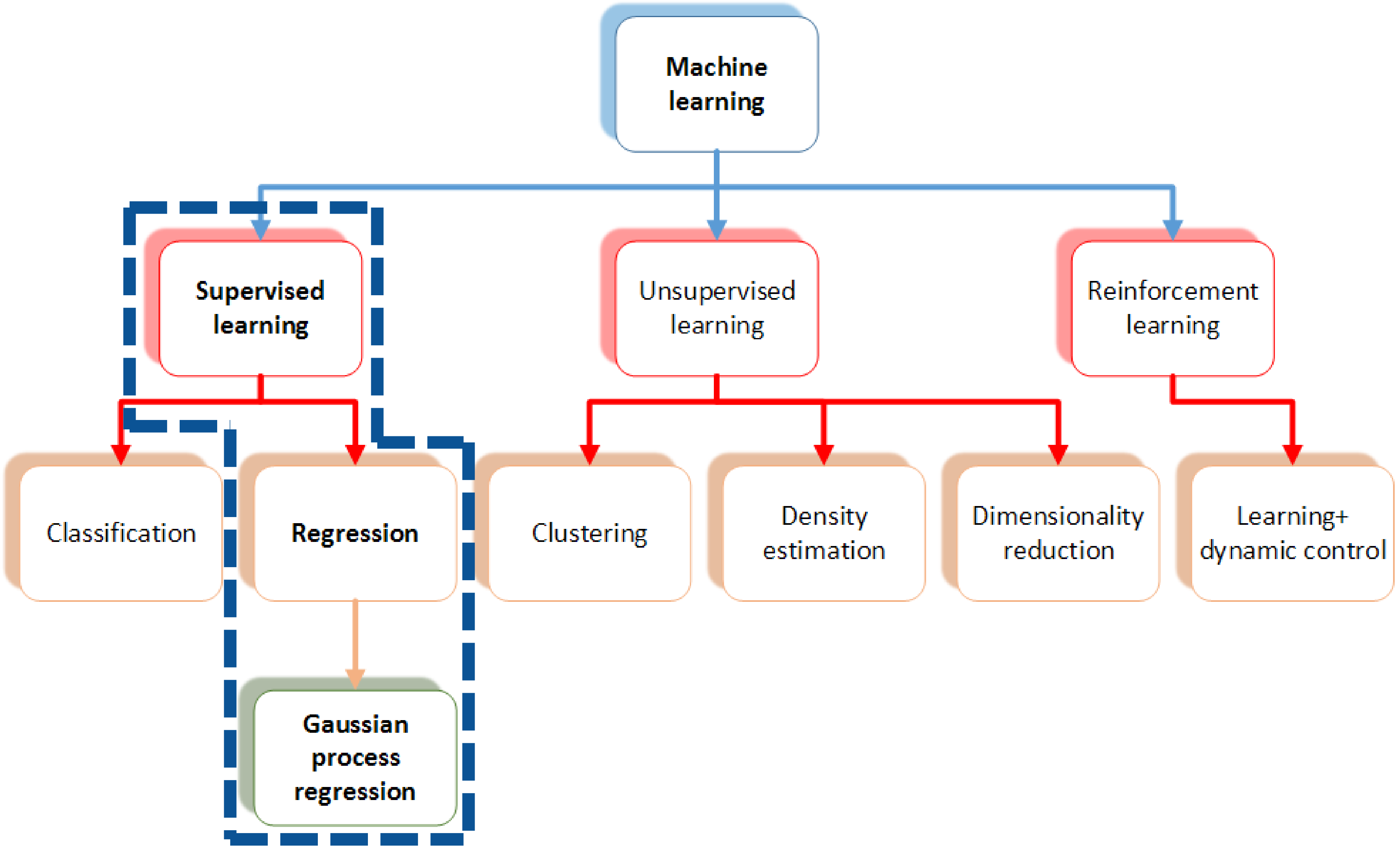
	64	128	256	512	1024
lin-1	< 0.01	< 0.01	< 0.01	0.01	0.02
knn-cv-1	< 0.01	0.04	0.15	1	6
mars3.6-bag-1	0.2	0.3	0.7	2	4
mlp-ese-1	2	4	8	16	32
gp-map-1	0.25	2	16	128	512
gp-mc-1	32	64	128	256	-
mlp-mc-1	32	64	128	256	512

- Summary
  - Among the approaches, Gaussian process regression (GPR)
    - Accuracy: outperform, robust, adaptive to the input.
    - Drawback: reaction rate is very slow.
  - Reason?
    - O(n^3) computational complexity.
    - O(n^2) storage complexity.

$\rightarrow$ Unacceptable for large-dataset computation.

Experiments from the publication: Rasmussen, C. E. (1996). *Evaluation of Gaussian processes and other methods for non-linear regression* (Doctoral dissertation, University of Toronto).

Taxonomy

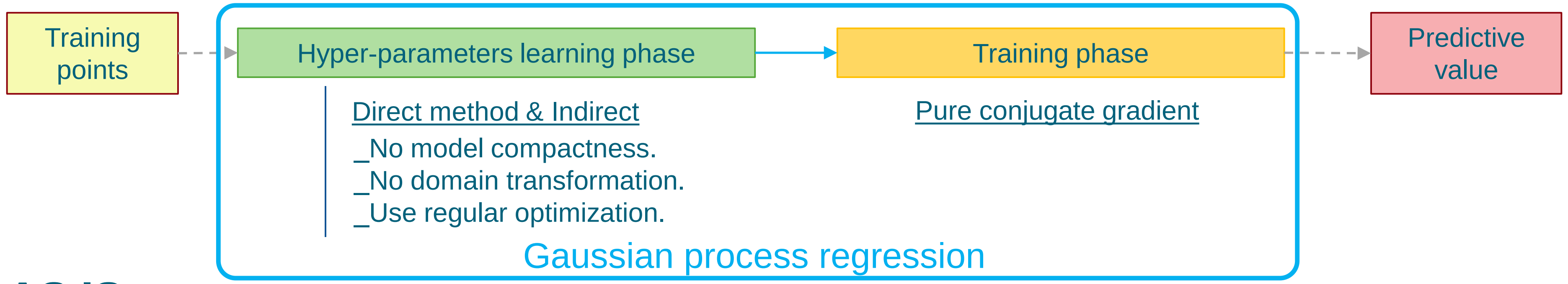

Machine learning
Supervised learning
Unsupervised learning
Reinforcement learning
Classification
Regression
Clustering
Density estimation
Dimensionality reduction
Learning+ dynamic control
Gaussian process regression

Research	Technique	Result	Limitation
[eGPR] - Evolving Gaussian process for predicting chaotic time series [1]	Conjugate Gradient + Cholesky decomposition.	O(n^2/6)	_Hyper-parameter learning phase only.
[nGPR] - Nonparametric adaptive control using Gaussian processes with online hyper-parameter estimation [2]	Entropy minimization + Kullback–Leibler divergence maximization	O(n^2)	_High computational complexity. _Sensitive to local optima. _Low accuracy
[lGPR] - Efficient Gaussian Process Regression for large data sets [3]	Random projection method + reduced rank matrix approx. → reduce dimensions of dataset	O(mn^2)	_High computational complexity.
[bGPR] - Gaussian process for Big Data [4]	Stochastic Variational Inference.	O(nm^2)	_Require additional parameters. _Low accuracy.
[oGPR] - Optimization of Gaussian process hyper-parameters using Rprop [5]	Rprop-gradient-based optimization technique (CG+quasi-Newton).	O(n^2)	_Hyper-parameter learning phase only.

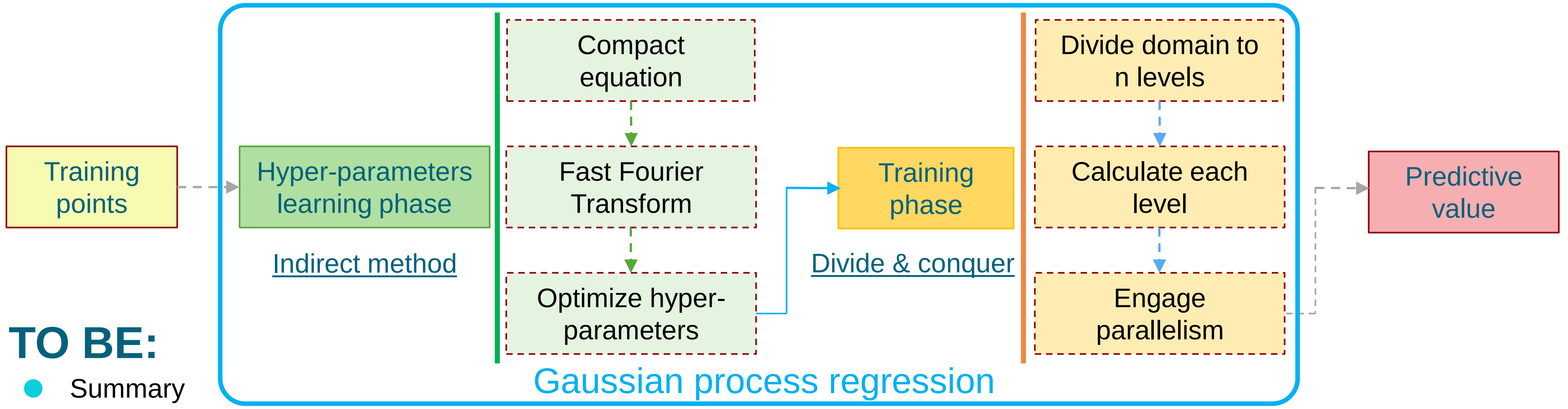
[1] Petelin, D., & Kocijan, J. (2014, June). Evolving Gaussian process models for predicting chaotic time-series. In *Evolving and Adaptive Intelligent Systems (EAIS), 2014 IEEE Conference on* (pp. 1-8). IEEE.
[2] Grande, R. C., Chowdhary, G., & How, J. P. (2013, December). Nonparametric adaptive control using gaussian processes with online hyperparameter estimation. In 52nd IEEE Conference on Decision and Control (pp. 861-867). IEEE.
[3] Banerjee, A., Dunson, D. B., & Tokdar, S. T. (2012). Efficient Gaussian process regression for large datasets. *Biometrika*, ass068.
[4] Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. arXiv preprint arXiv:1309.6835.
[5] Blum, M., & Riedmiller, M. A. (2013, April). Optimization of Gaussian process hyperparameters using Rprop. In ESANN.

[PE]

Related works | Summary



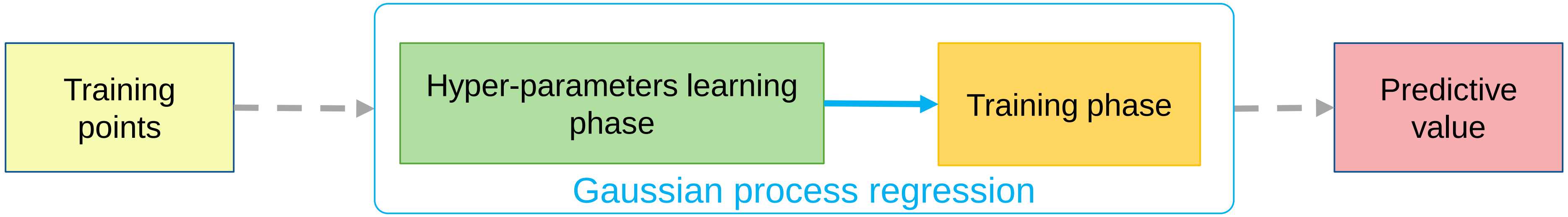
AS IS:



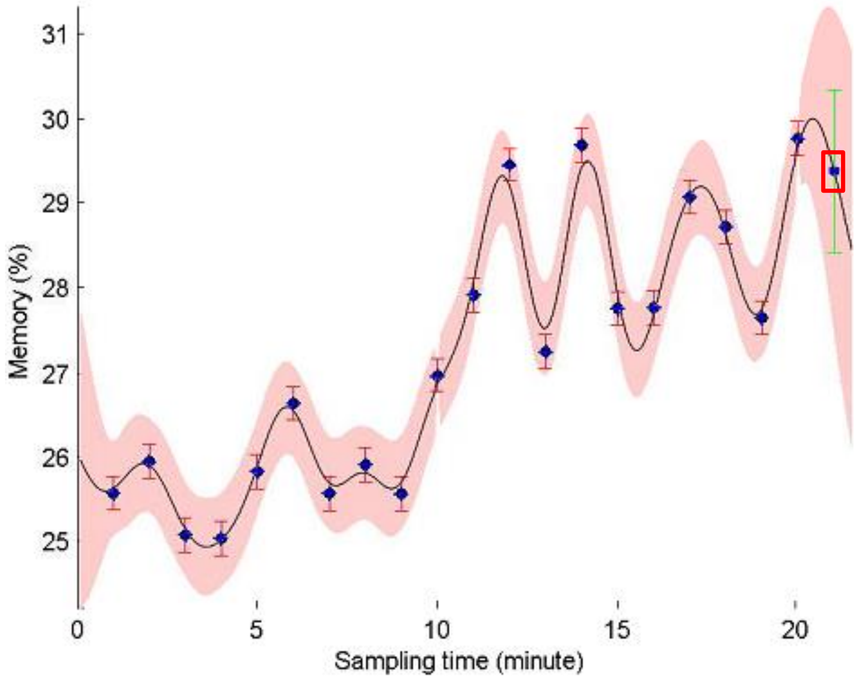
TO BE:

- Summary
  - Problem of existing works
    - High complexity.
    - Incomprehensive (apply only for hyper-parameters learning/training phase).
- Thesis proposes
  - Complexity reduction before prediction.
  - Comprehensively appropriate solution to both phases of GPR.
    - Improve HPLP by proposing the complexity reduction.
    - Improve Training phase by introducing “divide and conquer” idea.

Proposed method



- Thesis includes two phases
  - Hyper-parameters learning phase: adapts the parameters to the input data.
    - Usually, hyper-parameters located in kernel function.
    - Is used to build prediction model.
  - Training phase: does the regression based on the aforementioned hyper-parameters.
    - Less complex than hplp.
    - Usually performs much faster than hplp.
- Data source
  - Time location ***x***={x_1, x_2,…,x_n}.
  - Random variables ***y***={y_1, y_2,…,y_n}.
  - → Represent joint Gaussian distribution of target input data.
  - → The distribution overtime forms the GP



Predictive memory utilization based on 20 time-series training points collected from Ganglia.

$$f(y|x) \sim \mathcal{GP}(m(x), k(x,x')) \quad (2)$$

with

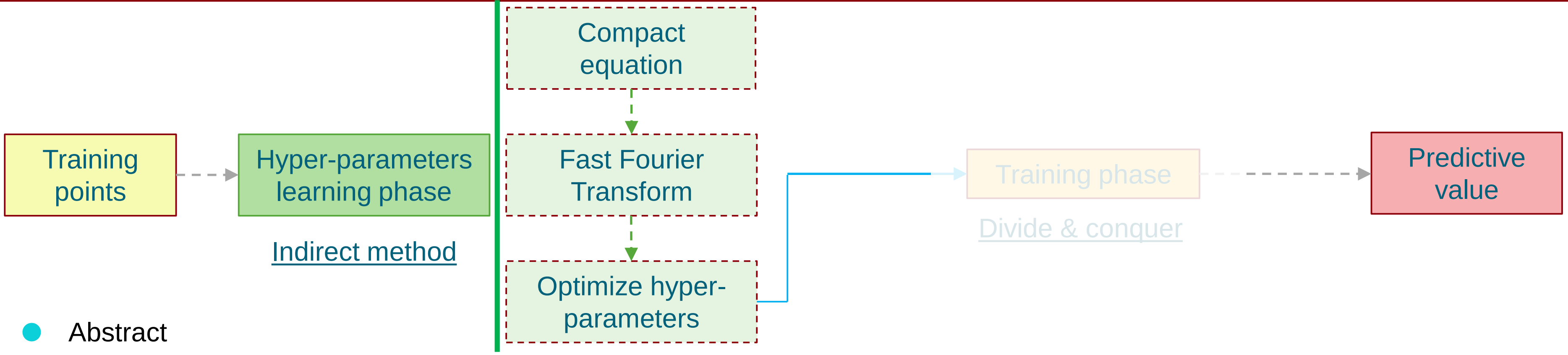
$$\boxed{m(x)} = \mathbb{E}(f(x)) \quad (3)$$

$$\boxed{k(x,x')} = \mathbb{E}\Big(\big(f(x)-m(x)\big)\big(f(x')-m(x')\big)\Big) \quad (4)$$

m(x) : mean function.

k(x,x') : kernel function, which consists of desired hyper-parameters.

Proposed method — Phase one: HPLP (1/7)



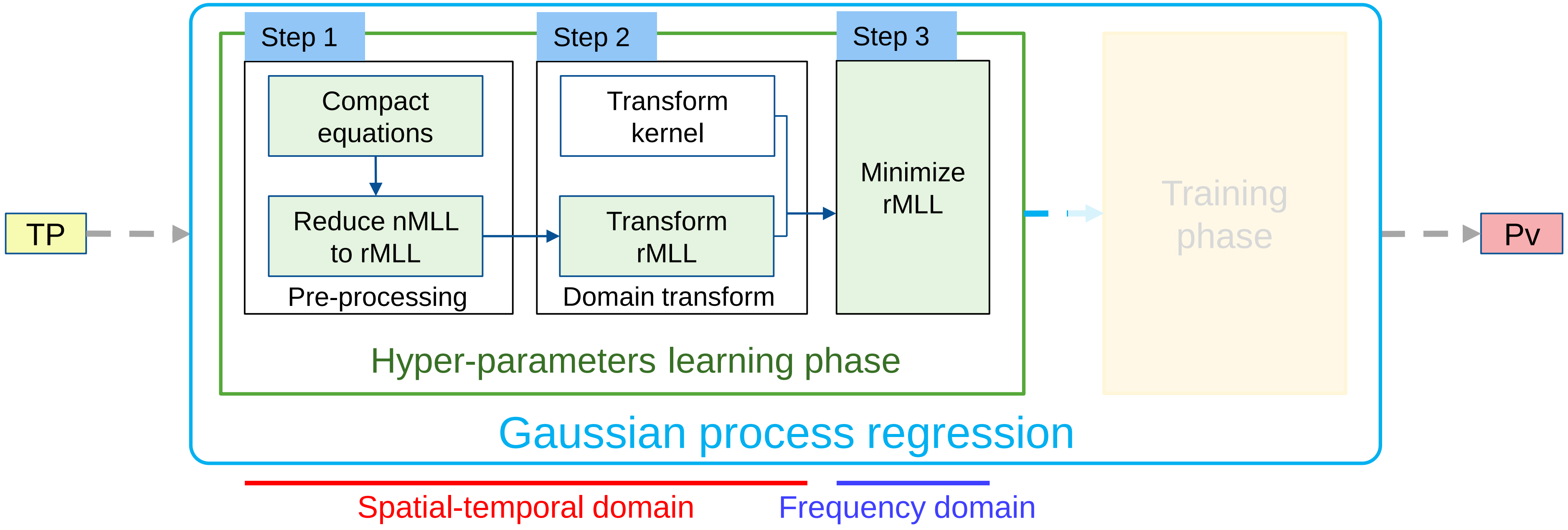
- Abstract
  - The proposed approach
    - Indirectly solves complexity by transforming the domain first, then optimize hyper-parameters.
      - Transforms problem from spatial-temporal domain to frequency domain.
      - Significantly reduces the computational complexity from O(n^3) to O($n$log$n$).
      - No need to store the inversed matrix on every step, reduce the storage complexity from O(n^2) to O(n).
    - Engage stochastic optimization to avoid local optima.
  - Background theories
    - Convergence law of log determinant.
    - Possibility of kernel transformation.
    - Effectiveness of Fast Fourier transform.
  - Benefit: cut down hyper-parameters learning time more than 82% [*].

Table 5.1: Computation cost of proposed method compared with others

	Direct method	eGPR	nGPR	bGPR	Proposed method
Hyper-parameters learning phase	$O(n^3)$	$O(n^2/6)$	$O(n^2)$	$O(nm^2)$	$O(nlogn)$
Training phase	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(m+n)$

[*] Bui, Dinh-Mao, Shujaat Hussain, Eui-Nam Huh, and Sungyoung Lee. "Adaptive Replication Management in HDFS Based on Supervised Learning." IEEE Transactions on Knowledge and Data Engineering 28, no. 6 (2016), pp: 1369-1382.

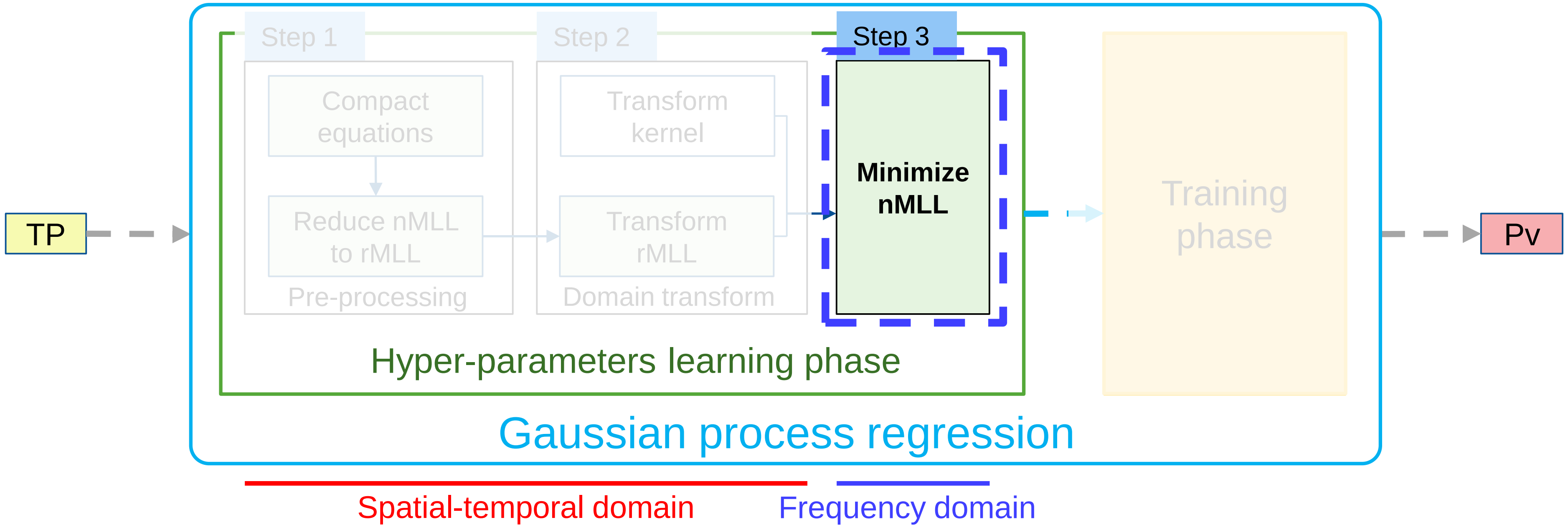
Proposed method — Phase one: HPLP (2/7)



- Proposed method: cooperation of
  - Fast Fourier transform
    - Transform the domain
    - Reduce the complexity to O($n$log$n$) while still maintain the integrity of the data.[*]
  - Convergence law of log determinant
    - Compact the negative marginal log likelihood.
    - Provide the convergence condition to preserve an acceptable accuracy level.
  - Stochastic gradient descent
    - Optimize the hyper-parameters.

[*] Schatzman, J. C. (1996). Accuracy of the discrete Fourier transform and the fast Fourier transform. *SIAM Journal on Scientific Computing*, *17*(5), 1150-1166.

Proposed method — Phase one: HPLP (3/7)



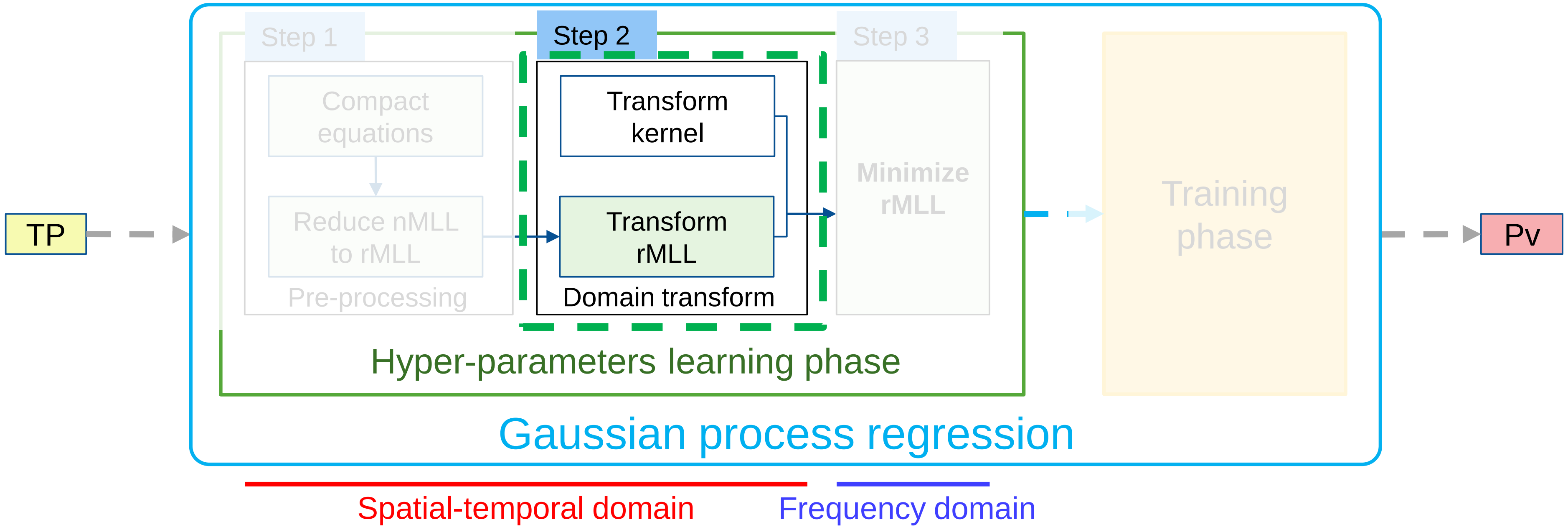
- Objective: find hyper-parameters to build covariance matrix.
- Traditional method: minimizing negative marginal log likelihood (nMLL)

$$-\log p(y|\theta) = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}| + \frac{n}{2}\log(2\pi) \qquad (5)$$

→Take partial derivatives on negative marginal log likelihood and set to zero.

- Result: high complexity in calculating
  - Matrix inverse
  - Partial derivative matrix
- Possible solution: domain transformation
  - Popular in mathematics.

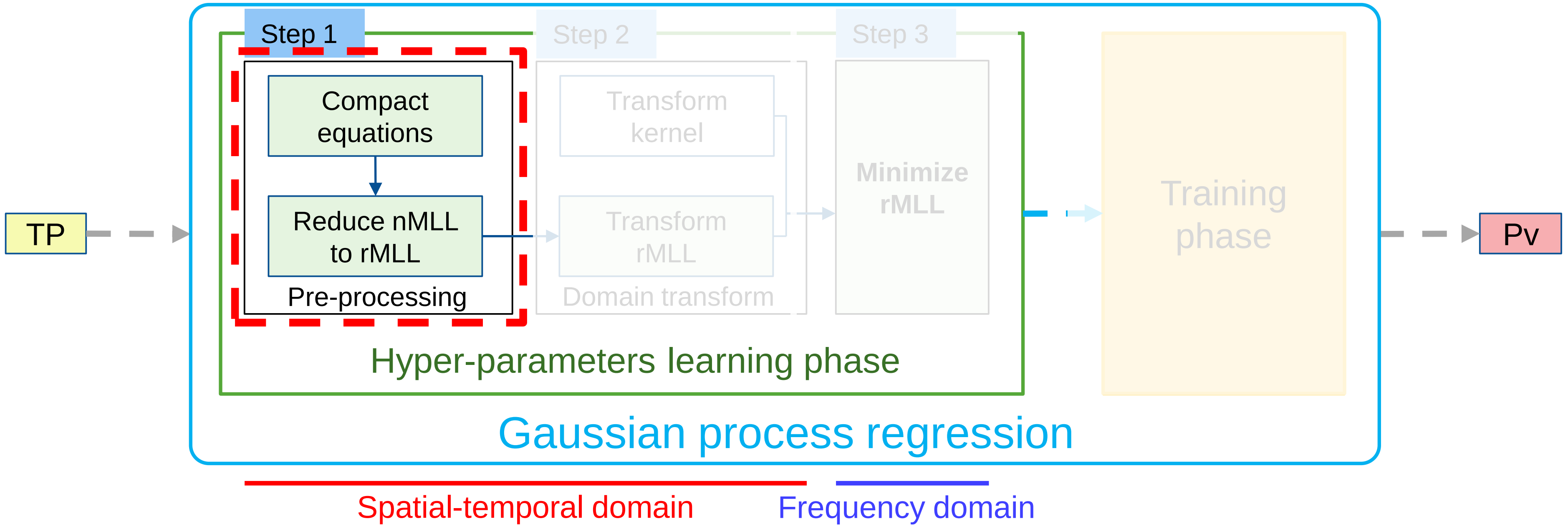
Proposed method | Phase one: HPLP (4/7)



- Observation
  - Negative marginal log likelihood consists of two components
    - Kernel function.
    - Negative marginal log likelihood itself.
- Domain transformation
  - Kernel transformation: is possible by engaging equivalent kernels [*]
  - Negative marginal log likelihood transformation: really complicated and infeasible.
    - Need a pre-processing step to compact the equations.
    - Make it possible to transform.

[*] Sollich, P. and Williams, C., 2005. Using the equivalent kernel to understand Gaussian process regression. In Advances in Neural Information Processing Systems (pp. 1313-1320).

Proposed method — Phase one: HPLP (5/7)



- Observation
  - Most of works focus on compacting negative marginal log likelihood (**nMLL**) to reduced version (**rMLL)**.
- Compact equations
  - Behind the scene: nMLL is reduced by compacting below corresponding equation

$$-\log p(y|\theta) = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}| + \frac{n}{2}\log(2\pi) \quad (6)$$

- After compactness and transformation, the minimization process is engaged to calculate the optimal hyper-parameters.
  - Stochastic Gradient Descent.

Proposed method

Phase one: HPLP (6/7)

- Step 1: compact the equation based on convergence law of log determinant [*].
- Step 2: transform the domain from spatial-temporal to frequency.

nMLL — Log determinant — Constant

$$-\log p(y|\theta) = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \boxed{\frac{1}{2}\log|\mathbf{K}|} + \boxed{\frac{n}{2}\log(2\pi)} \quad (7)$$

Preserve

Estimation

$$\boxed{\hat{L} = \log|\hat{\mathbf{K}}| - \tau_n} \quad (8)$$

Substitute

Drop

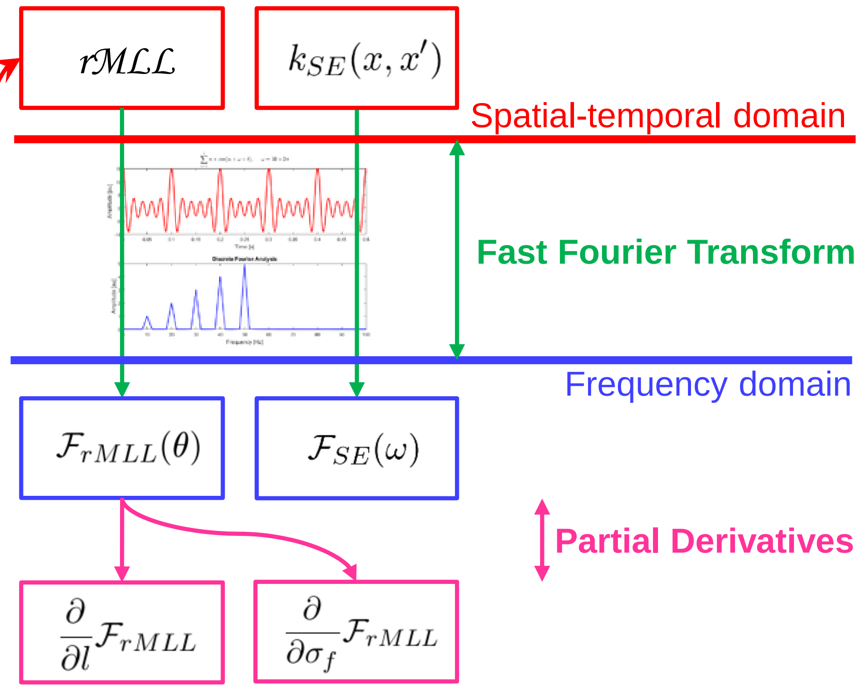
$$-\log p(y|\theta) = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \boxed{\frac{1}{2}\hat{L}} + \boxed{\frac{n}{2}\log(2\pi)} \quad (9)$$

Preserve

Convergence law

$$\boxed{-\log p(y|\theta)_{rMLL} = \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y}} \quad (10)$$

$\mathcal{rMLL}$



$\mathcal{rMLL}$
$k_{SE}(x,x')$
Spatial-temporal domain
Fast Fourier Transform
Frequency domain
$\mathcal{F}_{rMLL}(\theta)$
$\mathcal{F}_{SE}(\omega)$
Partial Derivatives
$\frac{\partial}{\partial l}\mathcal{F}_{rMLL}$
$\frac{\partial}{\partial \sigma_f}\mathcal{F}_{rMLL}$

- Computational complexity: O(n^3) to O($n\log n$).
- Storage complexity: O(n^2) to O(n) because no need to save the inversed matrix in each step.

[*] Cai, T. Tony, Tengyuan Liang, and Harrison H. Zhou. "Law of log determinant of sample covariance matrix and optimal estimation of differential entropy for high-dimensional Gaussian distributions." Journal of Multivariate Analysis 137 (2015): 161-172.

[**] Sollich, P. and Williams, C., 2005. Using the equivalent kernel to understand Gaussian process regression. In Advances in Neural Information Processing Systems (pp. 1313-1320).

Phase one: HPLP (7/7)

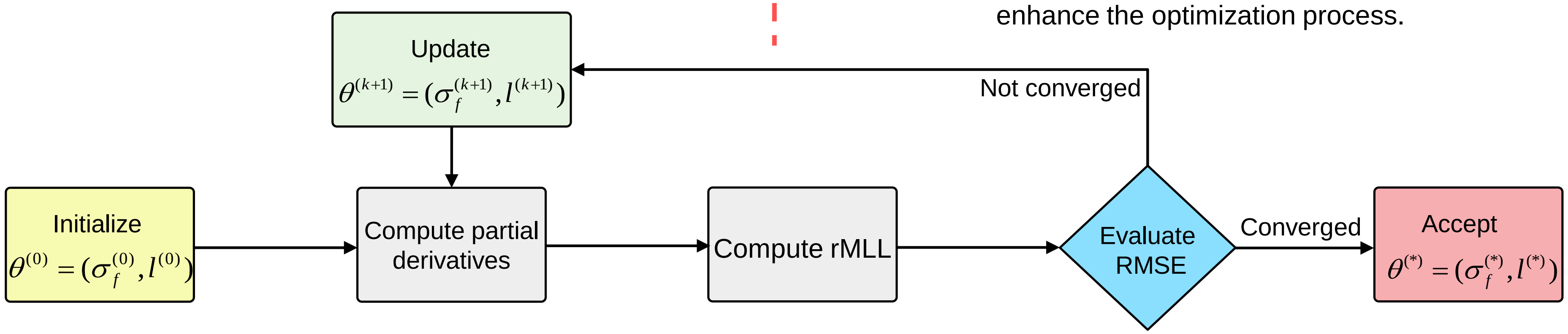
- Step 3: compute the hyper-parameters based on stochastic gradient descent (SGD)

$$l^{(k)} \leftarrow l^{(k-1)} + \alpha(k)\frac{\partial}{\partial l^{(k-1)}}\mathcal{F}_{rMLL} \quad (11)$$

$$\sigma_f^{(k)} \leftarrow \sigma_f^{(k-1)} + \alpha(k)\frac{\partial}{\partial \sigma_f^{(k-1)}}\mathcal{F}_{rMLL} \quad (12)$$

- Decay function [*]: $\alpha(k) = {}^{1}/_{(k+1)}$ (13)
- Stop condition: $RMSE = \sqrt{\frac{(\sum_{i=1}^{n}\mathcal{F}_i^{(k)} - \mathcal{F}_i^{(k-1)})^2}{n}}$ (14)
- SGD is less sensitive to local optima than the conjugate gradient. Thus, this technique can produce more accurate result.

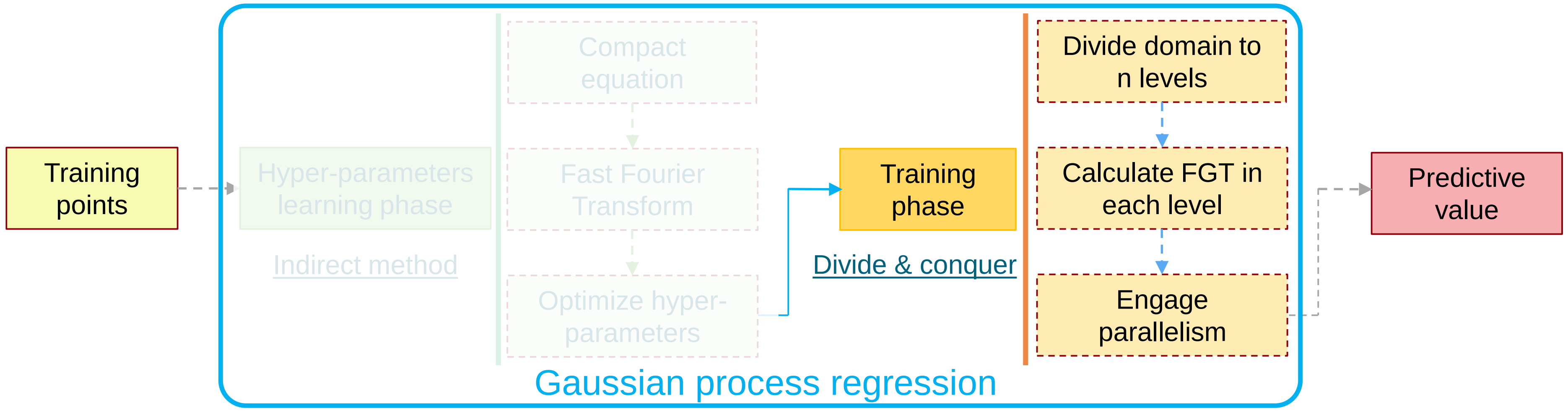
- Summary after phase one
  - The improvement has been made due to
    - Mathematical compactness of equation
    - Data transformation
    - Effective optimization
  - By this improvement
    - The hyper-parameters can be found faster than the regular conjugate gradient technique which is mostly used most in other methods ([eGPR], [bGPR]).
    - Less sensitive to the local optima, more accuracy [nGPR].
    - No extra-parameters requirement ([bGPR]).
  - The author's contribution
    - The mathematical derivations to compact and transform the marginal negative log likelihood.
    - The appropriate engagement and adjustment to enhance the optimization process.


Update
$\theta^{(k+1)} = (\sigma_f^{(k+1)}, l^{(k+1)})$
Not converged
Initialize
$\theta^{(0)} = (\sigma_f^{(0)}, l^{(0)})$
Compute partial derivatives
Compute rMLL
Evaluate RMSE
Converged
Accept
$\theta^{(*)} = (\sigma_f^{(*)}, l^{(*)})$

[*] Robbins-Monroe decay function is engaged to vanish the partial derivatives. This function helps to prevent the optimizing values jumping out of the designated boundary. Decay function is used instead of the exact line search or backtracking line search mainly due to the performance issue.

Proposed method

Phase two: training (1/5)


Training points
Hyper-parameters learning phase
Indirect method
Compact equation
Fast Fourier Transform
Optimize hyper-parameters
Training phase
Divide & conquer
Divide domain to n levels
Calculate FGT in each level
Engage parallelism
Predictive value
Gaussian process regression

- Abstract
  - Other conventional approaches
    - Only use iterative gradient methods.
    - Directly calculate the matrix inverse.
  - The proposed approach
    - Transforms the equation from matrix-vector product into sum of conjugacy.
    - Engages "divide and conquer" idea.
    - Reduces the computational complexity from O(n^2) to O(m+n).
  - Theoretical basis
    - Improved fast Gauss transform technique.
    - The mechanism of MapReduce.
  - Benefit: cut down training time more than 54.23% [*].

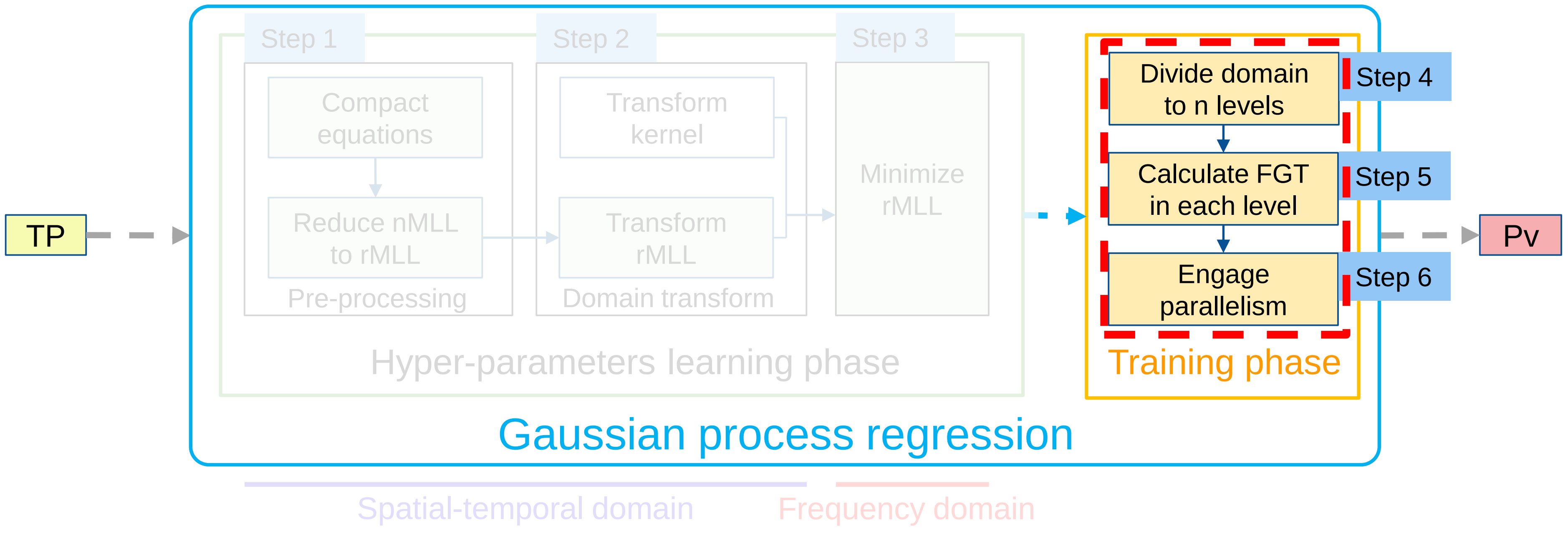
Table 5.1: Computation cost of proposed method compared with others

	Direct method	eGPR	nGPR	bGPR	Proposed method
Hyper-parameters learning phase	$O(n^3)$	$O(n^2/6)$	$O(n^2)$	$O(nm^2)$	$O(nlogn)$
Training phase	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(m+n)$

[*] Bui, Dinh-Mao, Shujaat Hussain, Eui-Nam Huh, and Sungyoung Lee. "Adaptive Replication Management in HDFS Based on Supervised Learning." IEEE Transactions on Knowledge and Data Engineering 28, no. 6 (2016), pp: 1369-1382.

Proposed method

Phase two: training (2/5)



- Objective: estimate the value of test input: y_*

$$\overline{y}_* = \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1} y \qquad (15)$$

- Rewrite equation

$$\overline{y}_* = \mathbf{K}(x_*, x)\xi \qquad (16)$$

where vector $$\xi = \mathbf{K}(x, x')^{-1} y \qquad (17)$$

multiply both sides of (17) by **K**(*x*,*x'*).

$$y = \boxed{\mathbf{K}(x, x')}\xi \qquad (18)$$

Covariance matrix can be retrieved by using previous hyper-parameters

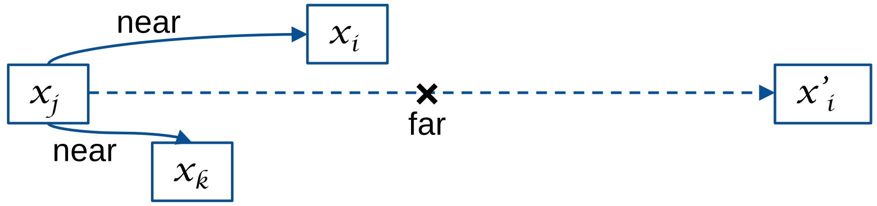
- Equation (16) is the target. To achieve that, equation (17) needs to calculate first.
- Since **K**(*x*,*x'*) is symmetric and positive-definite, ξ can be found by solving equation (18) by using linear conjugate gradient.
- Complexity: O(n^2).

Proposed method — Phase two: training (3/5)

$\mathbf{K}p_j$
Conjugate gradient

Transform element

$$\mathbf{K}p_j \triangleq G(x_j) = \sum_{i=0}^{N-1} q_i \boxed{w_j} \sigma_f \exp\left(\frac{-|x_j - x_i|^2}{2l^2}\right)$$

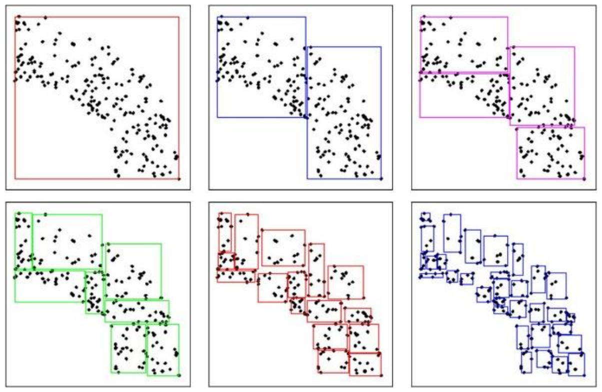


Discrete Gauss Transform

$O(N^2)$

$O(MN)$[*]

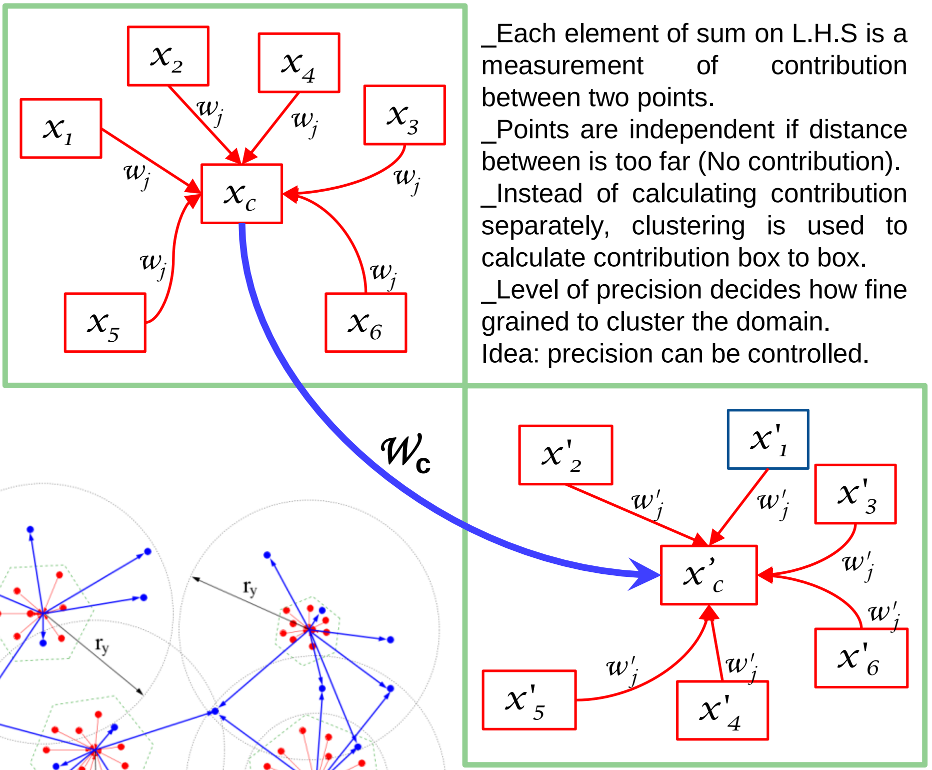
◆ Divide stage



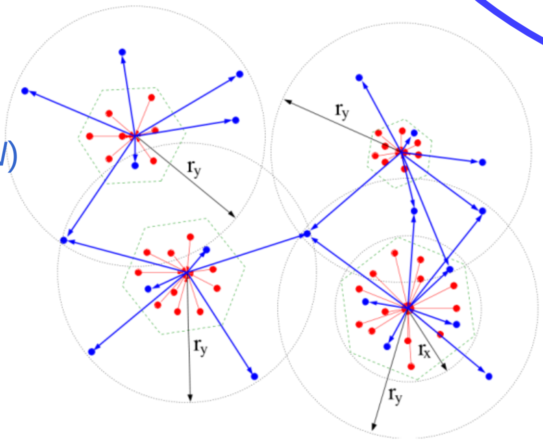
Fast Gauss Transform

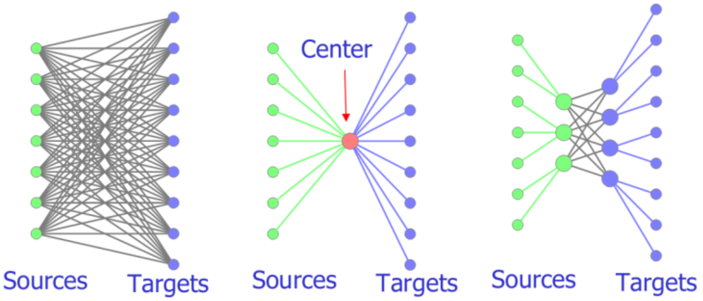
Step 4

$O(M+N)$



_Each element of sum on L.H.S is a measurement of contribution between two points.
_Points are independent if distance between is too far (No contribution).
_Instead of calculating contribution separately, clustering is used to calculate contribution box to box.
_Level of precision decides how fine grained to cluster the domain.
Idea: precision can be controlled.





[*] *M* is the number of target points and *N* is the number of source points.
[**] L. Greengard and J. Strain. The fast Gauss transform. SIAM J. Sci. Statist. Comput., 12(1): pp79–94, 1991.

Proposed method

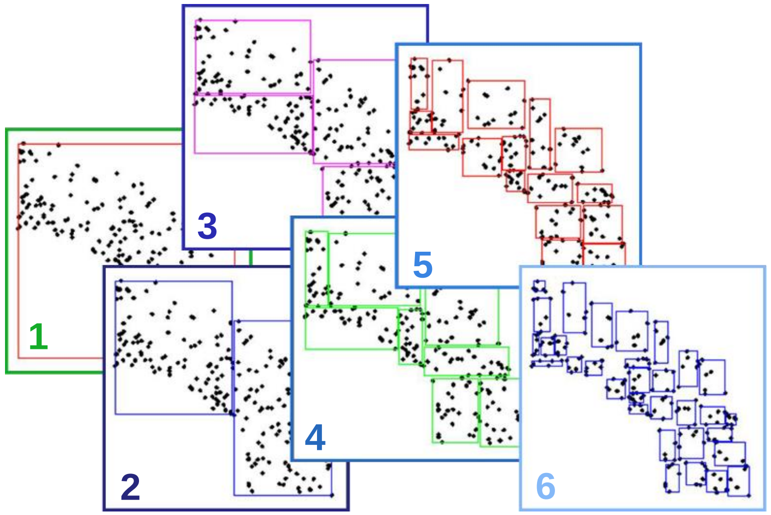
Phase two: training (4/5)

$$\mathbf{K}p_j \triangleq G(x_j) = \sum_{i=0}^{N-1} q_i \boxed{w_j} \sigma_f \exp\left(\frac{-|x_j - x_i|^2}{2l^2}\right)$$

in which:

q_i : adaptive weighted parameter related to the current level of precision ε.

w_j : wrapped coefficient propagated from neighbors.

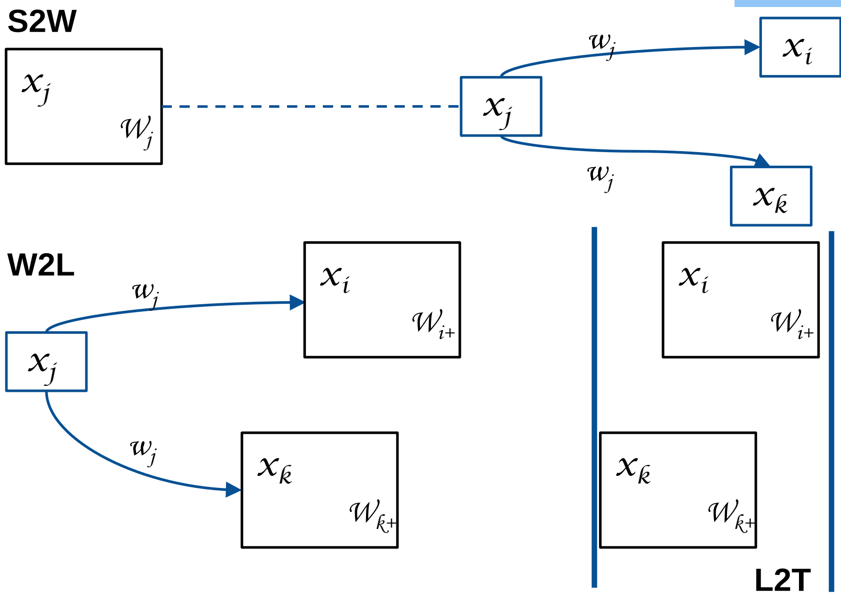


coarse to fine grained levels

Note: Adaptive weighted parameter q_i in Improve Fast Gauss Transform [*] are used to preserve the accuracy.

- **Conquer stage**: FGT consists of three sub-steps
  - **S2W**: calculate coefficient $\boxed{w_j}$ for each unit *U*, then propagate to neighbors of *U*.
  - **W2L**: update $w_j \longrightarrow w_{j+}$ based on the propagated values.
  - **L2T**: calculate new $\mathbf{K}p_j$ in the neighbor square based on the updated w_j

Step 5



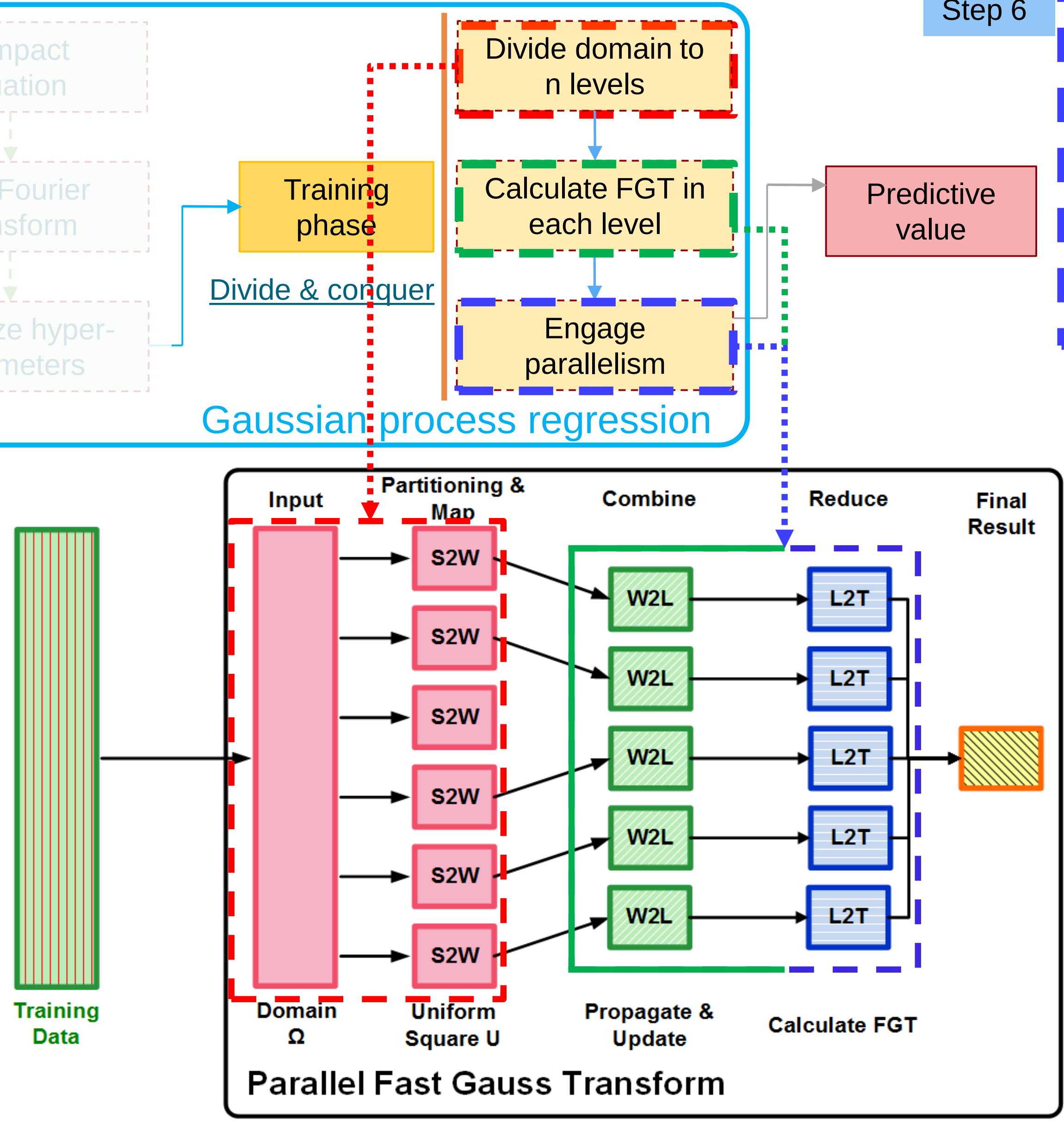
- Computational expense: O(*M*+*N*), linear. [**]

[*] Yang, C., Duraiswami, R., and Davis, L. Efficient kernel machines using the improved fast Gauss transform. In Advances in Neural Information Processing Systems. pp1561 -1568, 2005. Size of box is 2\sqrt{l ln(1/ε)}.

[**] T. Feder and D. Greene. Optimal algorithms for approximate clustering. InProc. 20th ACM Symp. Theory of computing, pages 434–444, Chicago, Illinois, 1988. Farthest point clustering [O(nlogk) for k-centers] compared with kmc [O(n$^{(dk+1)}$)]

Proposed method

Phase two: training (5/5)



Step 6

- **Parallelism**
  - The idea of MapReduce.
    - S2W to Map phase
    - W2L to Combine phase
    - L2T to Reduce phase
  - Calculate the $(Kq)_j$ for each level in parallel.
  - Further enhance the performance.

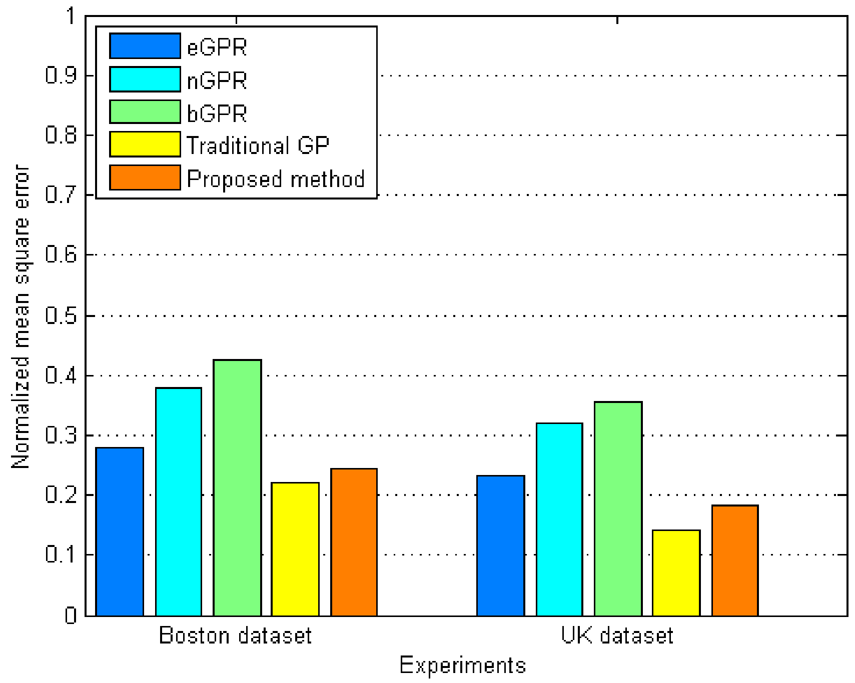
- Summary after phase two
  - The improvement has been made due to
    - Elimination of matrix inverse.
    - Transformation from matrix-vector product into sum of elemental DGT.
    - Speed up DGT but still preserve the accuracy by engaging IFGT method.
    - Idea of utilizing the parallelism.
  - By this improvement, the training phase outperforms the iterative method using in other technique.
  - The author's contribution
    - Introduce "divide and conquer" philosophy and parallelism in reducing the complexity and speeding computation.

Performance evaluation — Bench-markings (1)

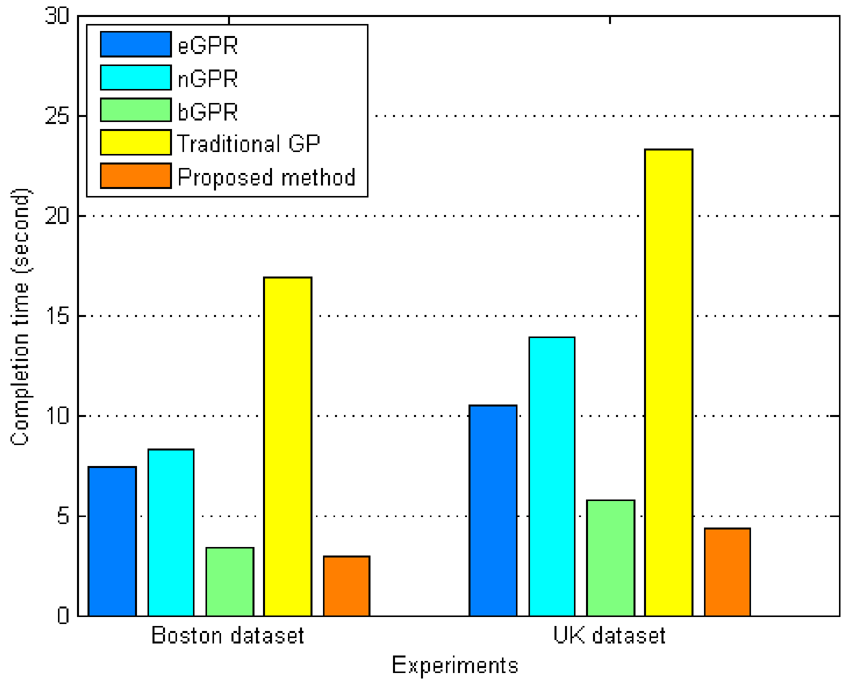
- Empirical experiment
  - Dataset
    - Boston housing dataset (small set)
      - Contains information collected by U.S Census Service concerning housing in area of Boston Mass.
      - Obtained from the StatLib archive (http://lib.stat.cmu.edu/datasets/boston)
      - Has been used extensively by the machine learning community for regression purpose.
      - Small size: 506 entries.
      - The median value of home is the attribute of interest.
    - UK land registry price paid dataset (large set)
      - Monthly price paid data in UK (http://data.gov.uk/dataset/land-registry-monthly-price-paid-data/).
      - Period of extraction: February to October 2012.
      - Covered area: England and Wales.
      - Large size: 75,000 entries.
      - Randomly select 8,000 entries for testing.
      - Divide into tenfold, each fold comprises 800 entries.
      - Results are averaged.
      - The changing cost of apartments is the attribute of interest.
  - Candidate algorithms [RWs]
    - eGPR
    - nGPR
    - bGPR
    - Traditional GP (GP-MAP)
    - Proposed method
  - Experiment metrics
    - Accuracy (measured by NMSE)
    - Speed (measured by CPU time).
  - Experiment facility
    - CPU: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz (8 CPUs), ~3.4GHz
    - Memory: 16GB
    - Operating System: Windows 7 Ultimate 64-bit

Performance evaluation | Bench-markings (1)

Accuracy (lower NMSE, better accuracy)



CPU time (lower is better)



- Accuracy measurement
  - Most accuracy: traditional GP (GP-MAP)
    - No degradation factor
  - Second best: the proposed method
    - Slightly degradation due to approx.
    - The loss of accuracy is controlled by IFGT.
    - Degradation: 2%→7% depends on dataset.
  - Worst accuracy: bGPR
  - More data increases the accuracy of the algorithms.
- Speed measurement
  - Fastest: proposed method
    - Cut down more than 82% of CPU time (compared with Traditional GP).
  - Second best: bGPR
  - Slowest: Traditional GP (GP-MAP)
- Summary
  - Proposed method: tradeoff between accuracy and speed.
  - Accuracy degradation is controlled.

Performance evaluation — Bench-markings (2)

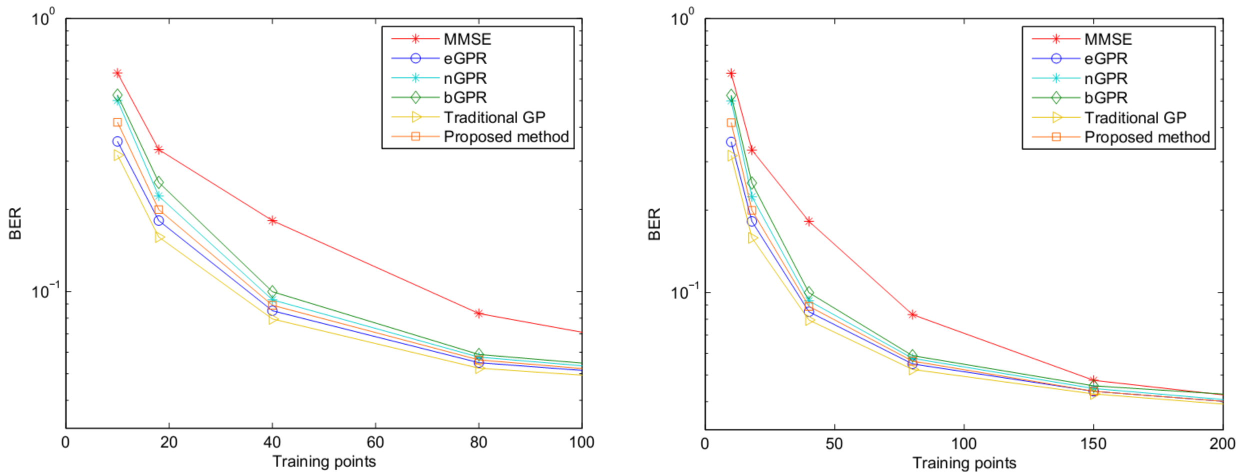


Figure 6.3: Relationship between BER and the number of training points

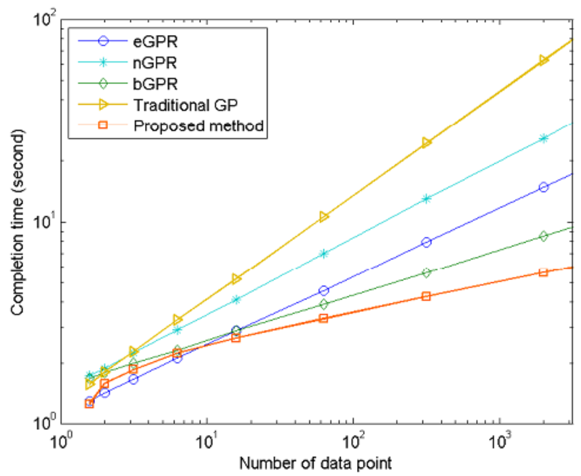


Figure 6.4: Time complexity enhancement of the proposed method.

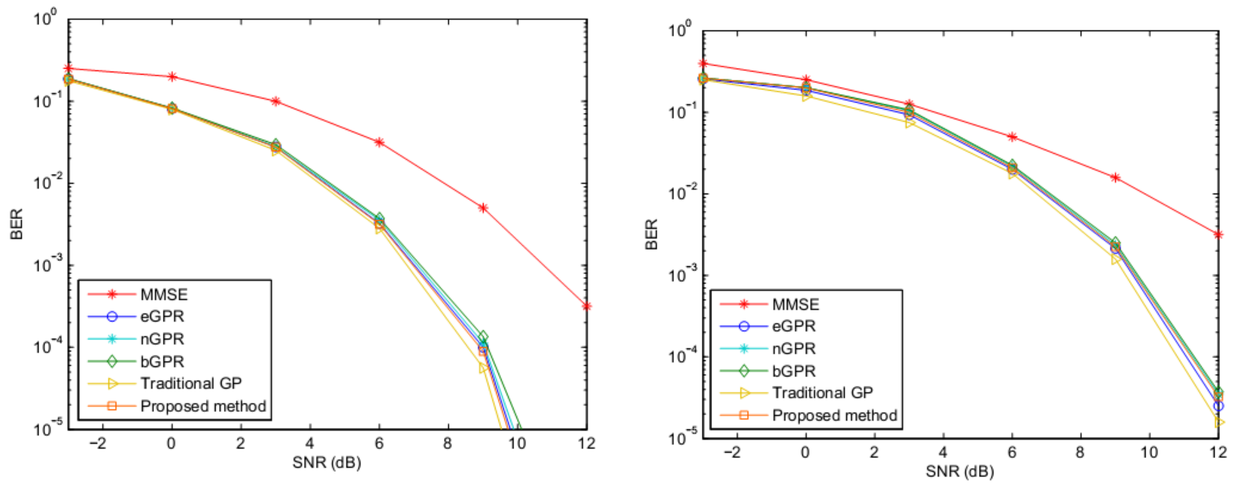


Figure 6.5: Bit-error-rate at each signal-to-noise ratio

- Synthetic experiment: Multiuser detection in DS-CDMA
  - Target system: synchronous DS-CDMA.
  - Number of user: 8 users spreading by Gold sequences.
  - Length of binary sequences: 31.
  - Power of all users: SNR= 4dB.

- Accuracy measurement
  - Little degraded but closely approach the Traditional GP (under 2%).
  - More training points improve the accuracy.
- Speed measurement
  - Much outperforms the Traditional GP.

Bui, Dinh-Mao, and Sungyoung Lee. "Fast Gaussian Process Regression for Multiuser Detection in DS-CDMA." IEEE Communications Letters (2016).

Potential applications (1)

- Energy efficiency for cloud computing system [*].
  - The fact: power used for turning-off and turning-on PMs is much less than maintaining idle PMs.
  - **Problem-to-solve**: achieving energy savings while ensuring the quality of services.
  - Power saving logic
    1. Monitoring component (Ganglia) collects desired system statistics (**SS**).
    2. **SS** is provided as input to the **Predictor** to produce predictive statistics (**PS**)
    3. **PS** is engaged in **Energy Optimizer** to create near optimal number (**NON**) of hosting physical machines w.r.t quality of services.
    4. Based on **NON**, power manager creates **VM Migrating Instruction** to consolidate VMs.
    5. Size of pool of active PMs is closely reduced to **NON**. Idle PMs are shut down to save power consumption.
    - Proposed method is applied to the prediction technique enhance the usefulness of monitoring data.
  - Measurement: various.
  - Quality of services: according to SLA document.
  - Dataset: Google traces [**]

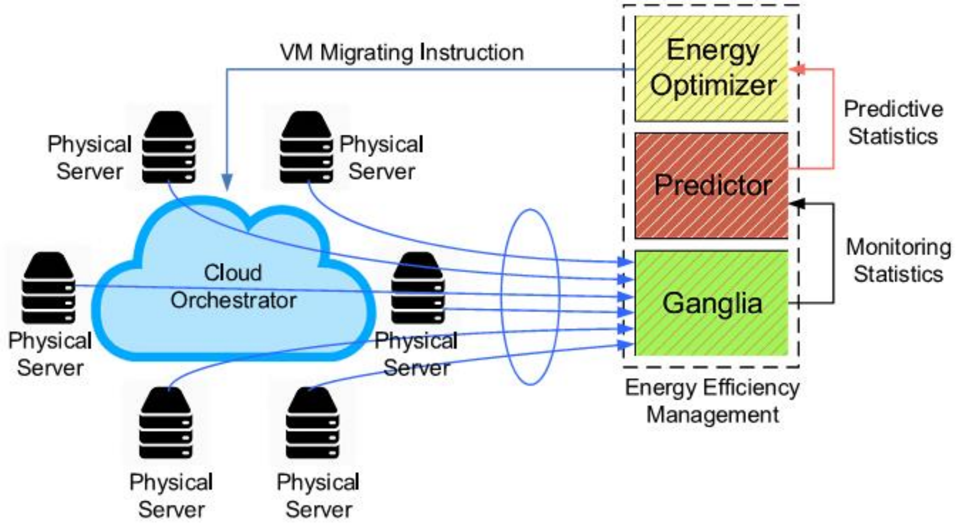


Fig. 2. Architecture of energy efficiency management (E2M) system.

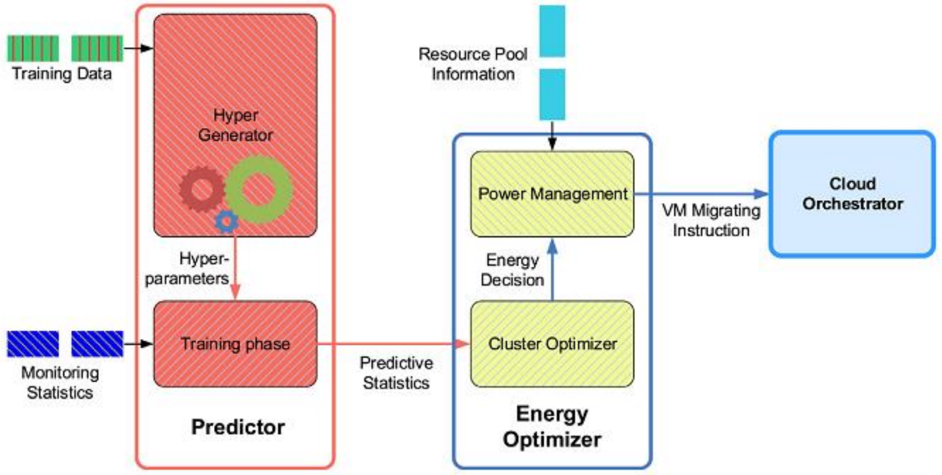
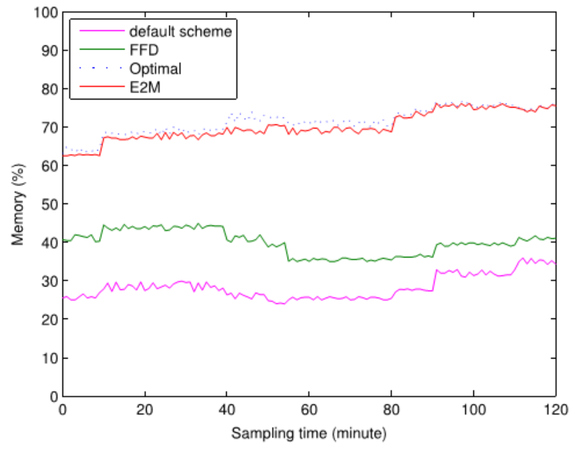


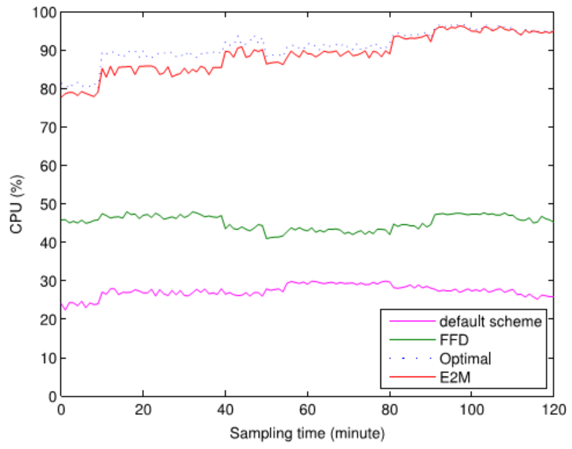
Fig. 3. Flowchart of energy efficiency management (E2M) system.

Bui, Dinh-Mao, YongIk Yoon, Eui-Nam Huh, SungIk Jun, and Sungyoung Lee. "Energy efficiency for cloud computing system based on predictive optimization." Journal of Parallel and Distributed Computing, Vol 102, April, 2017, pp: 103-114.
[**] https://github.com/google/cluster-data/

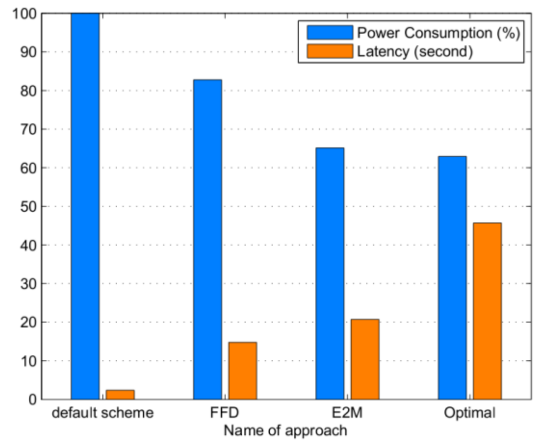
Performance evaluation | Potential applications (1)



(a) Average memory utilization (higher is better).



(b) Average CPU utilization (higher is better).



(a) Evaluation on Google traces experiment.

Fig. 4. Utilization evaluation of the proposed method in Google traces experiment.

- Experiment design
  - Metrics:
    - CPU & Memory utilization.
    - System latency.
    - Power consumption.
    - Prediction accuracy
  - Algorithms in experiments
    - Default scheme: no power saving.
    - Greedy first fit decreasing (FFD).
    - Proposed approach (E2M).
    - Optimal scheme: optimize energy only.
  - Dataset: choose the biggest homogeneous set (6732 PMs).

Table 6.2: Google cluster's organization and configuration

Quantity of nodes	Category	CPU	RAM
1	1	0.50	0.06
3	3	1.00	0.50
5	1	0.50	0.97
5	1	0.50	0.03
52	1	0.50	0.12
126	2	0.25	0.25
795	3	1.00	1.00
1001	1	0.50	0.75
3863	1	0.50	0.25
6732	1	0.50	0.50

Table 6.3: Summary of Google traces' characteristics

Time span	# of PMs	#VM requests	Trace size	# of users
29 days	12583	>25M	>39GB	925

- Result - utilization analysis
  - Metrics: CPU & Memory utilization.
  - Benchmark
    - Lowest: **default scheme**.
    - Highest: **optimal scheme**.
    - 2nd highest: **E2M**
  - Reason
    - **Default scheme**: too many idle PMs → low utilization.
    - **Optimal scheme**: too much focus on energy saving regardless system overhead.
    - **E2M**: focus on energy saving w.r.t system latency. (54.72% less latency than optimal scheme.)

Performance evaluation | Potential applications (1)

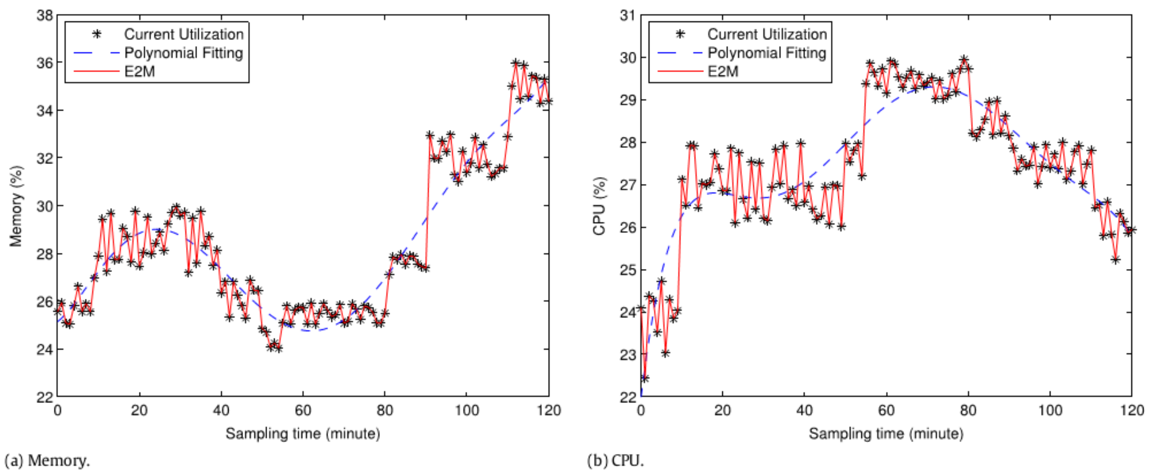


Fig. 10. Prediction evaluation of the proposed method in Google traces experiment.

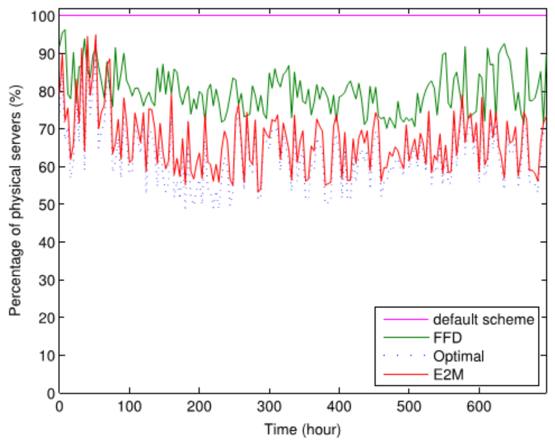


Fig. 5. Percentage of active physical servers in Google traces experiment.

- Result - power consumption analysis
  - Default scheme: burns huge amount of electricity.
    - Keeping all machines running even without any workload.
  - FFD: burns less than default scheme but still wastes energy.
    - Leave many PMs under-utilized.
    - Unable to flexibly balance the resource pool at high workload period.
  - Optimal scheme: performs pre-calculated optimal scheme to achieve great energy saving (37.08%).
    - Does not consider system overhead→ very high utilization.
    - Tend to violate quality of services.
  - Proposed scheme: achieve near-optimal energy saving scheme (34.89%), but proactively control system utilization.
- Result - enhancement of monitoring statistics
  - Proposed method outperform polynomial fitting interpolation.
  - Provide better input for optimization scheme.

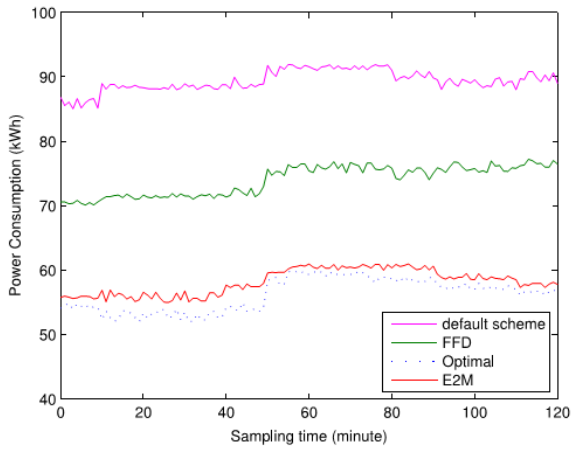
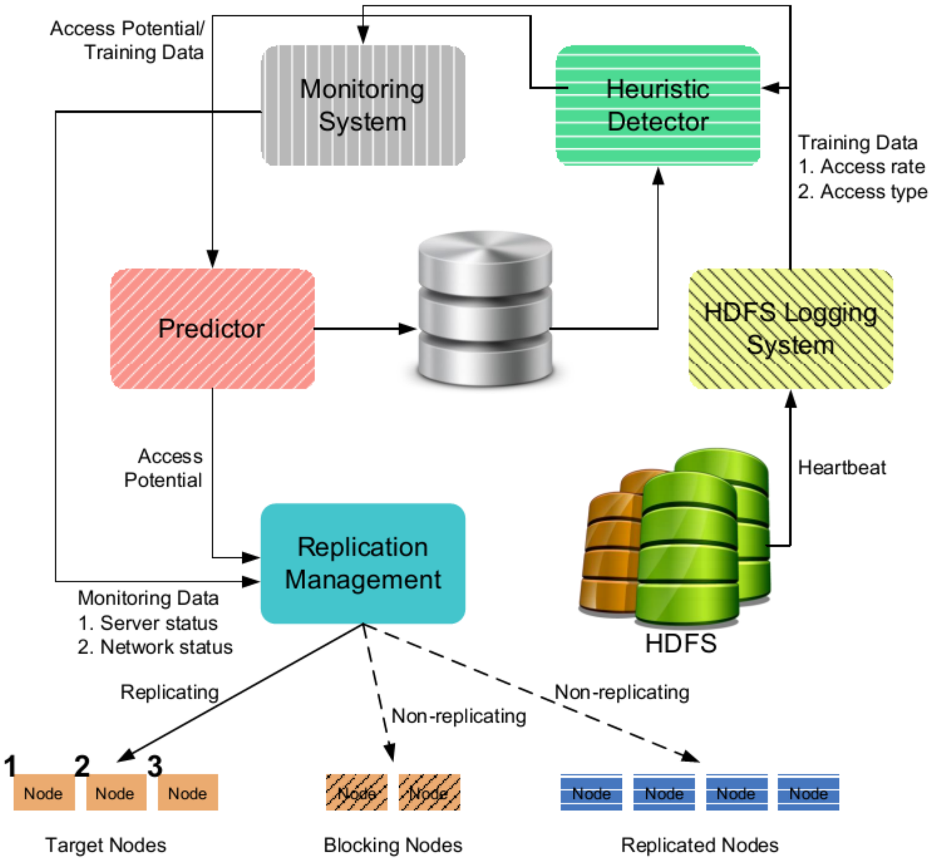


Fig. 6. Power consumption evaluation of the proposed method in Google traces experiment (lower is better).

Potential applications (2)

- Adaptive replication management in HDFS
  - The fact: over 80% data accesses go toward less than 10% of stored data.
  - **Problem-to-solve**: Identify and replicate the 10% active data with adaptive factor of replication.
    - Reduce overhead on data transfer and storage than default scheme (triplication)
    - Improving Map phase (especially long tasks) in Hadoop system by enhancing data locality.
  - Definitions
    - Access potential: the predictive rate that a specific file would be accessed in a given unit of time.
    - Data locality metric: the average ratio between local accesses and total accesses on HDFS files.
  - Replicating logic
    - Logging system collects **heartbeat** of HDFS, which consists of **access information** (time, rate (read/write), type(remote/local access)…). Heartbeat is cleaned and sent to predictor.
    - Predictor calculates **access potential** of files and sends to replication management.
    - Replication management calculates suitable **replication factor** based on **access potential** and assigns to corresponding files.

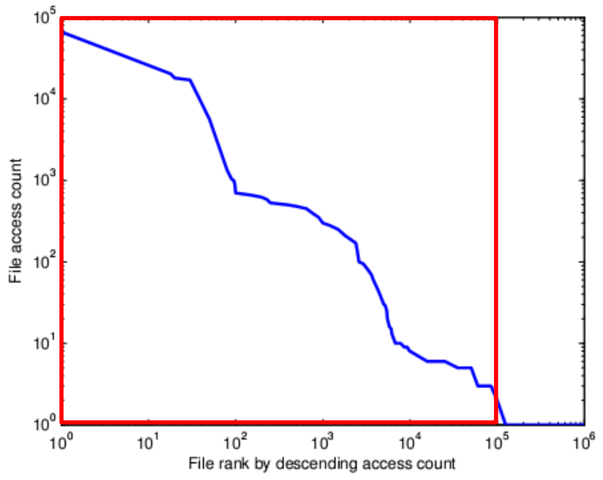


- Datasets:
  - Modified Statistical Workload Injector for MapReduce (SWIM) based on FB traces [*]
  - Built-in TeraSort developed by Yahoo!.

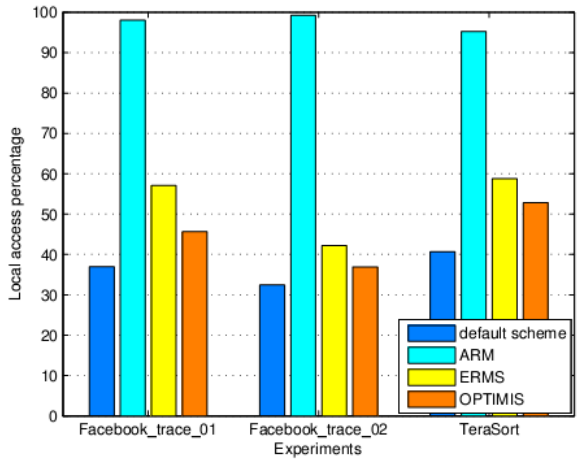
Bui, Dinh-Mao, Shujaat Hussain, Eui-Nam Huh, and Sungyoung Lee. "Adaptive Replication Management in HDFS Based on Supervised Learning." IEEE Transactions on Knowledge and Data Engineering 28, no. 6 (2016), pp: 1369-1382.
[*] https://github.com/SWIMProjectUCB/SWIM

Performance evaluation

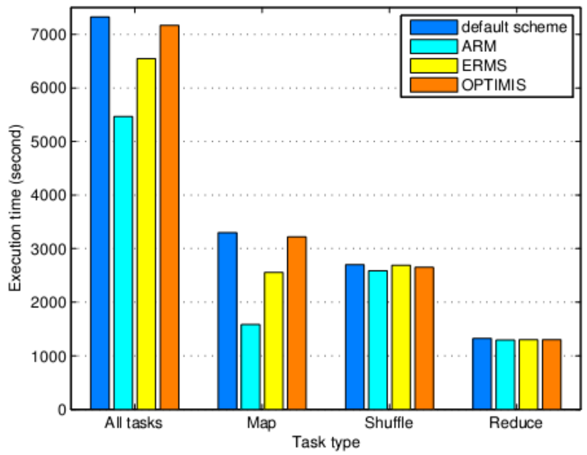
Potential applications (2)



(a) Access rate vs rank.



(e) Data locality metric evaluation (higher is better).



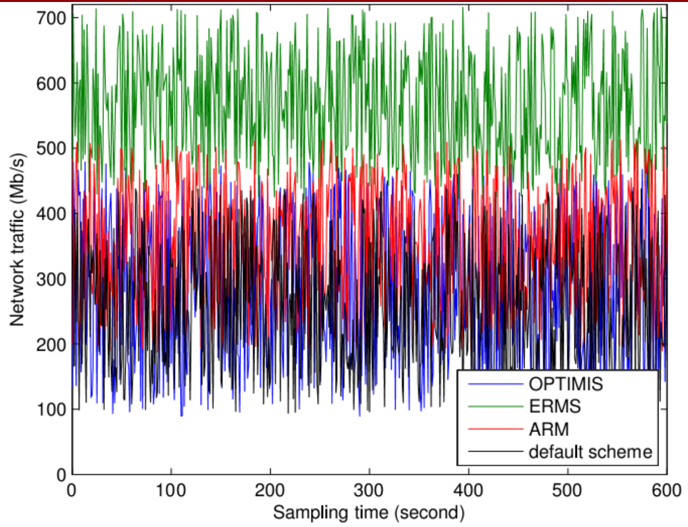
(f) Execution time benchmark in the TeraSort stress test (lower is better).

- Experiment design
  - Metrics:
    - Data locality.
    - Execution time.
    - Network traffic.
  - Algorithms in experiments
    - Default scheme: triplication.
    - ERMS.
    - OPTIMIS.
    - Proposed method (ARM)
  - Dataset:
    - Modified SWIM (FB traces): 24 historical traces sampled on 600-machines cluster following Zipf-like distribution. (FB-2009 & FB-2010).
    - TeraSort: performed on 1TB input data.

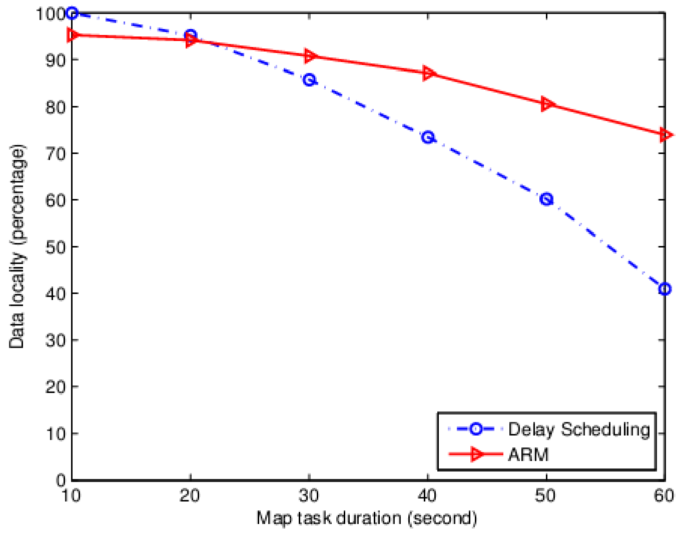
- Result - availability analysis
  - Metric: averaged data locality
  - Benchmark
    - Lowest: default scheme.
    - Highest: **proposed approach** (ARM, 2.84 times better than default scheme).
    - 2nd highest: ERMS
- Result - performance gain
  - Metric: execution time.
  - Benchmark: ARM reduces 52% and 37.98% mapping time compared with default scheme and second best approach (ERMS)
  - Notice: enhancement only takes effect on the Map task only.

Potential applications (2)

- Result - other observations
  - Network traffic evaluation
    - ERM achieves second best in availability *via* over-replication. ➔ delivers a lot of **overhead** to the network (39.96% higher than proposed approach – ARM).
    - OPTIMIS achieves low network contention due to under-replication strategy ➔ limits data locality.
    - Proposed approach – **ARM** has **similar network load** compared to default scheme (triplication) but **achieves much better** data locality.
    - Reason: ARM chooses right set of active data to replicate.
  - Data locality vs Map-task durations
    - TeraSort benchmark: size of task increase gradually.
    - Compared with **state-of-the-art** - delay scheduling algorithm (delay-sched).
    - Result
      - Map-task duration < 20s: delay-sched achieves 4.71% better data locality than ARM.
      - Map-task duration = 22s: delay-sched and ARM score analogous result.
      - Map-task duration > 22s: ARM step by step overcome delay-sched.
- Summary
  - Proposed approach (ARM) identifies right set of active data and appropriately increases corresponding replication.
  - ARM deals well with long Map-tasks. In case of short tasks, delay scheduling algorithm is better.



(d) Network traffic evaluation (lower is better).



(g) ARM vs delay scheduling in the data locality test over various task durations.

Conclusion & Achievements

- In this thesis, the PhD candidate
  - Thoroughly analyze the nature of Gaussian process regression with regard to popularly used prediction techniques.
  - Propose the complexity reduction in hyper-parameter learning phase.
  - Introduce the “divide and conquer” idea into algorithm design in training phase.
- Quantitative improvement
  - The propose method outperforms other methods and achieves the significant improvement (more than 82% compared with traditional GP) in terms of completion time.
  - The degradation of accuracy is controlled under 7%, which is acceptable in most of GP applications.
- Future works
  - Engage parallel computing to the hyper-parameters learning phase.
  - Improve the proposed method to deal with heterogeneous system.
  - Consider extending the proposed method to the high-dimensional space.

- **SCI Journals – First author**
  - IEEE Transactions on Knowledge and Data Engineering (IF: 3.438, **published**, 2016).
  - IEEE Communications Letters (IF:1.988, **published**, 2017).
  - Elsevier journal of parallel and distributed computing (IF:1.93, **published**, 2017).
  - Journal of supercomputing (IF:1.326, **published online**, 2017).
  - Applied Intelligence (IF:1.215, **published**, 2015).
- SCI(E) Journals – Co-author
  - 2 MDPI Sensors.
  - IETE Review.
- Patents:
  - 2 patents in application.
  - 1 patent registered.
- Conferences
  - 6 international conferences.
  - 2 KCC conferences.

References

[1] C. E. Rasmussen, “Evaluation of gaussian processes and other methods for non-linear regression.” Ph.D. dissertation, Toronto, Ont., Canada, Canada, 1997, aAINQ28300.

[2] C. Rasmussen and C. Williams, Gaussian Processes for Machine Learning, ser. Adaptive Computation And Machine Learning. MIT Press, 2005. [Online]. Available: http: //www.gaussianprocess.org/gpml/chapters/

[3] D. Petelin and J. Kocijan, “Evolving gaussian process models for predicting chaotic time-series,” in Evolving and Adaptive Intelligent Systems (EAIS), 2014 IEEE Conference on. IEEE, 2014, pp. 1–8.

[4] R. Grande, G. Chowdhary, and J. How, “Nonparametric adaptive control using Gaussian processes with online hyperparameter estimation.” in Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on, Dec 2013, pp. 861–867.

[5] A. Banerjee, D. Dunson, and S. Tokdar, “Efficient Gaussian Process Regression for Large Data Sets.” ArXiv e-prints, Jun. 2011.

[6] J. Hensman, N. Fusi, and N. D. Lawrence, “Gaussian processes for big data.” CoRR, vol. abs/1309.6835, 2013.

[7] Y. Shen, A. Y. Ng, andM. Seeger, “Fast gaussian process regression using kd-trees.” in NIPS, 2005.

[8] C. Yang, R. Duraiswami, and L. S. Davis, “Efficient kernel machines using the improved fast gauss transform.” in NIPS, 2004.

[9] R. Beatson and L. Greengard, “A short course on fast multipole methods,” Wavelets, multi-level methods and elliptic PDEs, vol. 1, pp. 1–37, 1997.

[10] R. Kress, V. Maz’ya, and V. Kozlov, Linear integral equations. Springer, 1989, vol. 82.

[11] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, “Gaussian processes for time-series modeling.” Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 371, no. 1984, 2012.

[12] J. Cunningham, Z. Ghahramani, and C. E. Rasmussen, “Gaussian processes for time-marked time-series data.” in AISTATS, ser. JMLR Proceedings, N. D. Lawrence and M. Girolami, Eds., vol. 22. JMLR.org, 2012, pp. 255–263.

[13] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[14] M. E. Tipping, “Bayesian inference: An introduction to principles and practice in machine learning.” in Advanced Lectures on Machine Learning, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. Rtsch, Eds., vol. 3176. Springer, 2003, pp. 41–62. [Online]. Available: http://dblp.uni-trier.de/db/conf/ac/ml2003.html

[15] S. Brahim-Belhouari and J. Vesin, “Bayesian learning using gaussian process for time se-ries prediction.” in Statistical Signal Processing, 2001. Proceedings of the 11th IEEE Signal Processing Workshop on, 2001, pp. 433–436.

[16] G. Chowdhary, H. Kingravi, J. How, and P. Vela, “Bayesian nonparametric adaptive control using gaussian processes.” Neural Networks and Learning Systems, IEEE Transactions on, vol. PP, no. 99, pp. 1–1, 2014.

[17] M. A. Alvarez and N. D. Lawrence, “Computationally efficient convolved multiple output gaussian processes.” J. Mach. Learn. Res., vol. 12, pp. 1459–1500, Jul. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2021048

[18] D.-M. Bui, H.-Q. Nguyen, Y. Yoon, S. Jun, M. B. Amin, and S. Lee, “Gaussian process for predicting cpu utilization and its application to energy efficiency,” Applied Intelligence,vol. 43, no. 4, pp. 874–891, 2015.

[19] R. Gallager, Stochastic Processes: Theory for Applications. Cambridge University Press, 2013. [Online]. Available: http://books.google.co.kr/books?id=CGFbAgAAQBAJ

[20] D.-M. Bui, Y. Yoon, E.-N. Huh, S. Jun, and S. Lee, “Energy efficiency for cloud computing system based on predictive optimization,” Journal of Parallel and Distributed Computing, vol. 102, pp. 103–114, 2017.

[21] X. Fan, W.-D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized com-puter,” in ACM SIGARCH Computer Architecture News, vol. 35, no. 2. ACM, 2007, pp. 13–23.

[22] D. Meisner, B. T. Gold, and T. F. Wenisch, “Powernap: eliminating server idle power,” in ACM Sigplan Notices, vol. 44, no. 3. ACM, 2009, pp. 205–216.

[23] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, “Vmware distributed resource management: Design, implementation, and lessons learned,” Vmware Technical Journal, vol. 1, no. 1, pp. 45–64, 2012.

[24] I. Sarji, C. Ghali, A. Chehab, and A. Kayssi, “Cloudese: Energy efficiency model for cloud computing environments,” in Energy Aware Computing (ICEAC), 2011 International Con-ference on. IEEE, 2011, pp. 1–6.

[25] “What is ganglia?” http://ganglia.sourceforge.net/, accessed: 2015-08-13.

[26] A. Verma, P. Ahuja, and A. Neogi, “pmapper: power and migration cost aware application placement in virtualized systems,” in Middleware 2008. Springer, 2008, pp. 243–264.

[27] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling,” in Proceedings of the 5th European conference on Computer systems. ACM, 2010, pp. 265–278.

Thanks for listening.
Q&A?