# Potential of Applying kNN with Soft Walltime to Improve Scheduling Performance


**Thanh Hoang Le Hai,**
**Loc La Hoang, Nam Thoai**

thanhhoang@hcmut.edu.vn
High Performance Computing Laboratory
Advanced Institute of Interdisciplinary Science and Technology
Ho Chi Minh City University of Technology (HCMUT)
VNU Ho Chi Minh City

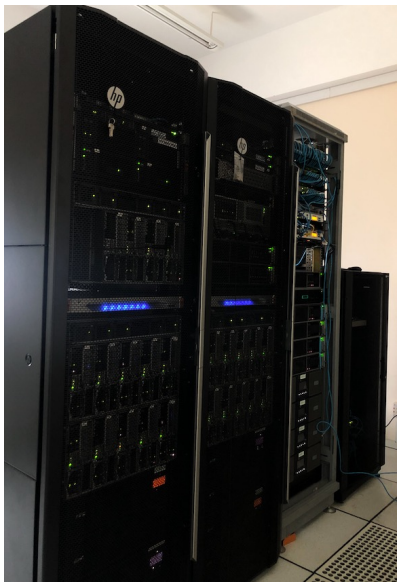
# Outline

- **Introduction**
  - Scheduling on HPC systems
  - User Walltime Prediction
- **Walltime correction using *kNN***
- **Soft Walltime Scheme**
- **Evaluation**
- **Conclusion and Future work**

# Scheduling on HPC Systems

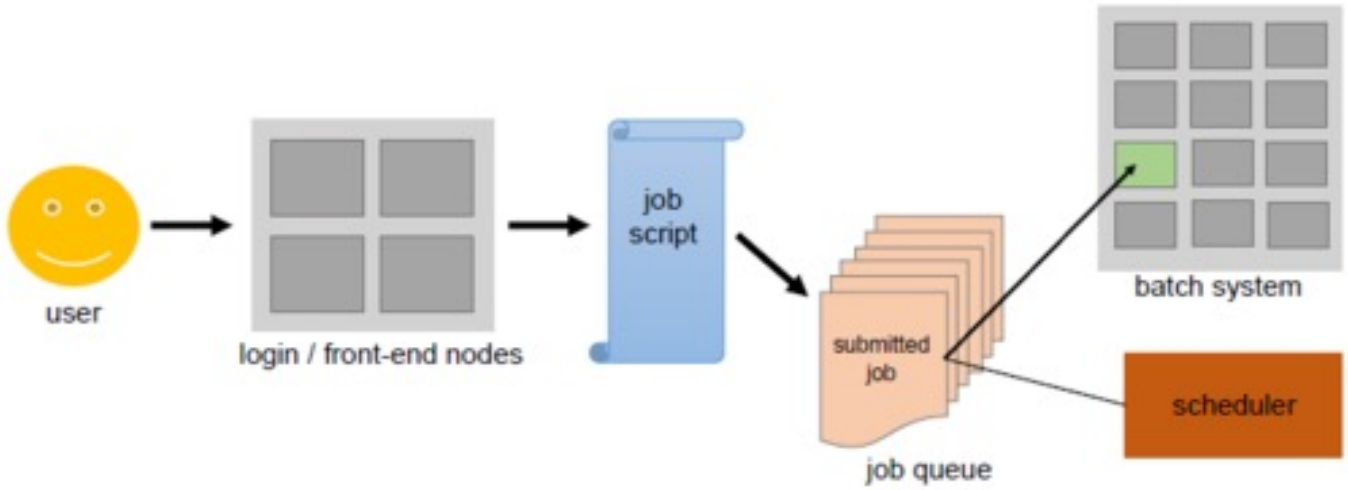
- Today, **High-Performance Computing** (HPC) is a key factor that leads to a significant number of scientific discoveries.

Photo: RIKEN



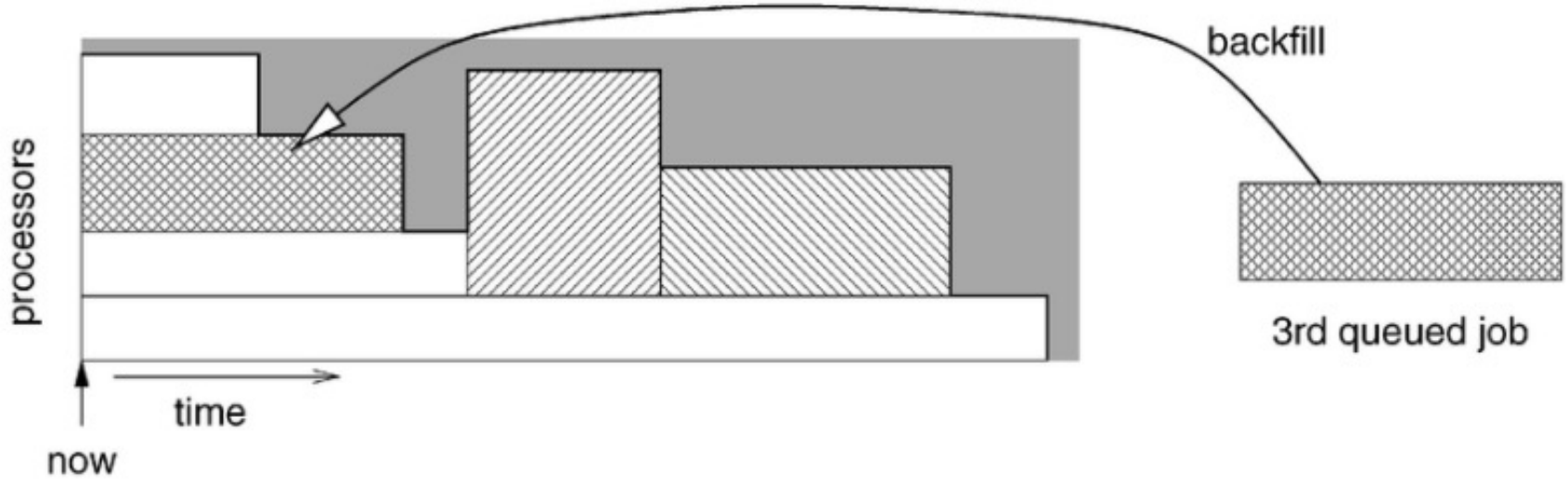
# Scheduling on HPC Systems

- **Resource and Job Management System** (RJMS) helps ensure fair access to computing resources while maintaining the optimal system utilization



# Scheduling on HPC Systems

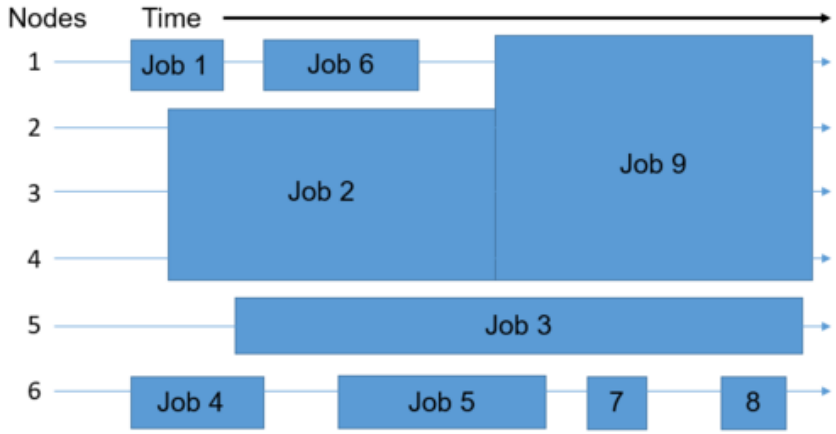
Job schedulers usually use a primary FCFS queue with **Backfilling**




**Source:** Ahuva Mu'alem Dror G. Feitelson, *"Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling", July 2001, IEEE Transactions on Parallel and Distributed Systems 12(6):529 - 543*

# Scheduling on HPC Systems

- Users may (or have to) provide their **walltime estimates** before submitting their workloads
- However, user predictions are usually **poor**



# User Walltime Prediction

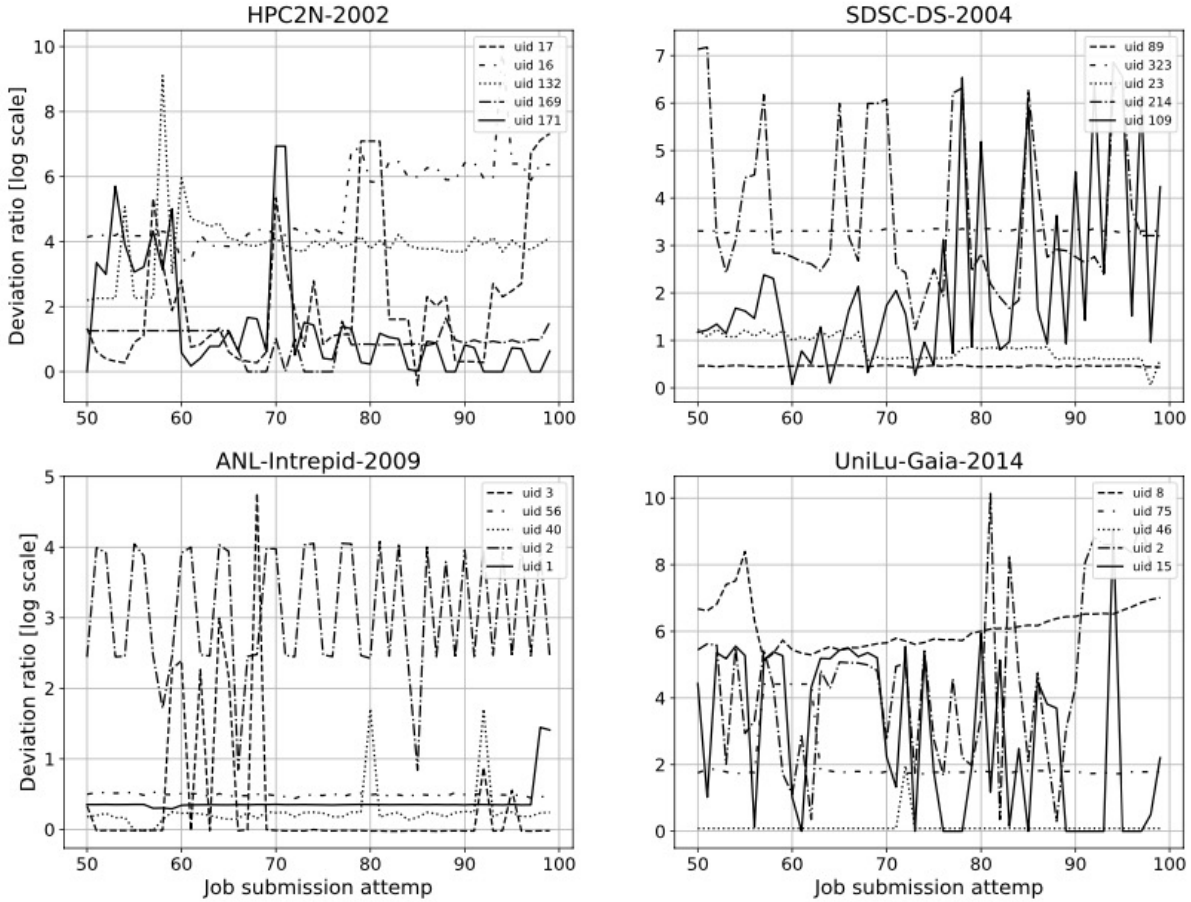
- Workload logs are collected from the **Parallel Workload Archive** (PWA)

STATISTICS OF FOUR WORKLOADS

| Workload Trace | From | Duration | #Jobs | #Nodes |
|---|---|---|---|---|
| HPC2N-2002 | Jul 2002 | 42 months | 202,871 | 240 |
| SDSC-DS-2004 | Mar 2004 | 13 months | 96,069 | 171 |
| ANL-Intrepid-2009 | Sep 2009 | 8 months | 68,936 | 640 |
| UniLu-Gaia-2014 | May 2014 | 3 months | 51,987 | 2,004 |

# User Walltime Prediction

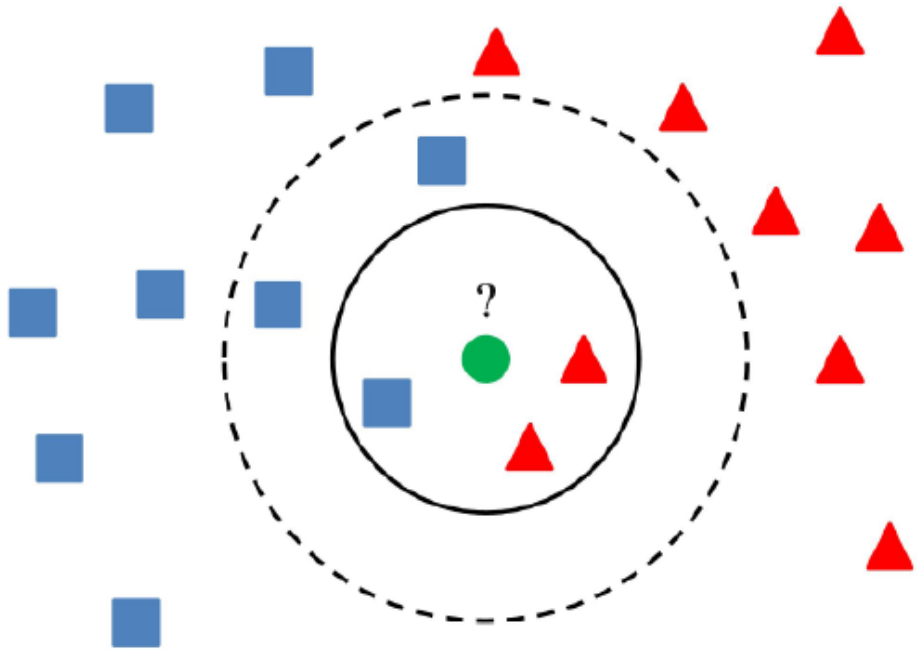
- User estimations are **inaccurate** although they already had some experience already



# Walltime correction using *kNN*

**Solution**: ***k*NN** method!

The inaccurate user estimate of a job is **refined** using the **historic data** about its most similar jobs



# Walltime correction using *kNN*

**Similarity =** Distance between points

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

**Label =** Major voting

$$q^c = \arg\max_{c \in C} \sum_{i=1}^{k} w_i \delta\left(c = \overline{T}_i^c\right)$$

# Walltime correction using *kNN*

However, some special attributes such as **uid** does not make sense in term of similarity!

$p$: finished job
$q$: new job

$$A = (j^r, j^u, j^w, \widetilde{j^w})$$

requested resources | uid | actual walltime | predicted walltime

$$d(p,q) = \sqrt{\sum_{i \in A'} (q^i - p^i)^2 + d_{pq}^u}$$

$A'$

$$d_{pq}^u = \begin{cases} \epsilon & p^u \neq q^u, \\ 0 & p^u = q^u. \end{cases}$$

ε: influence of historical information of the current user

# Walltime correction using *kNN*

After finding most similar items
calculate the **walltime correction factor** $\Delta_q$

$$w_i = e^{-\alpha d_i^{\beta}}$$

Exponential weight

$$\Delta_p = \frac{\widetilde{p^w}}{p^w}$$

Deviation ratio

$$\Delta_q = \begin{cases} \dfrac{\sum_i^k w_i \Delta_i}{\sum_i^k w_i} & |\overline{T}| > 0, \\ 1 & |\overline{T}| = 0. \end{cases}$$

$T$: Set of $k$ nearest neighbors **above the threshold** $\Theta$ **from** $N$ **latest finished jobs**

# Walltime correction using *kNN*

Finally, calculate the **refined** walltime

Predicted walltime

$$q^w = \frac{\widetilde{q^w}}{\Delta_q}$$

User provided walltime

Predicted deviation ratio

# Soft Walltime Scheme

- **Underestimated** walltime might terminate uncompleted tasks and make HPC systems unreliable.
- **Soft Walltime** scheme from OpenPBS use predicted values only for making reservation decisions
- The original **hard walltime** from users is still treated as the **upper bound of execution time**.

**Safe and Clear to users!**

# Evaluation

- Use **Batsim** with four selected workload
- Compare the predicted walltime with:
  - Perfect estimates
  - No correction
  - The per-user basis method *min.diff* (min diff of last 5 same user submissions)

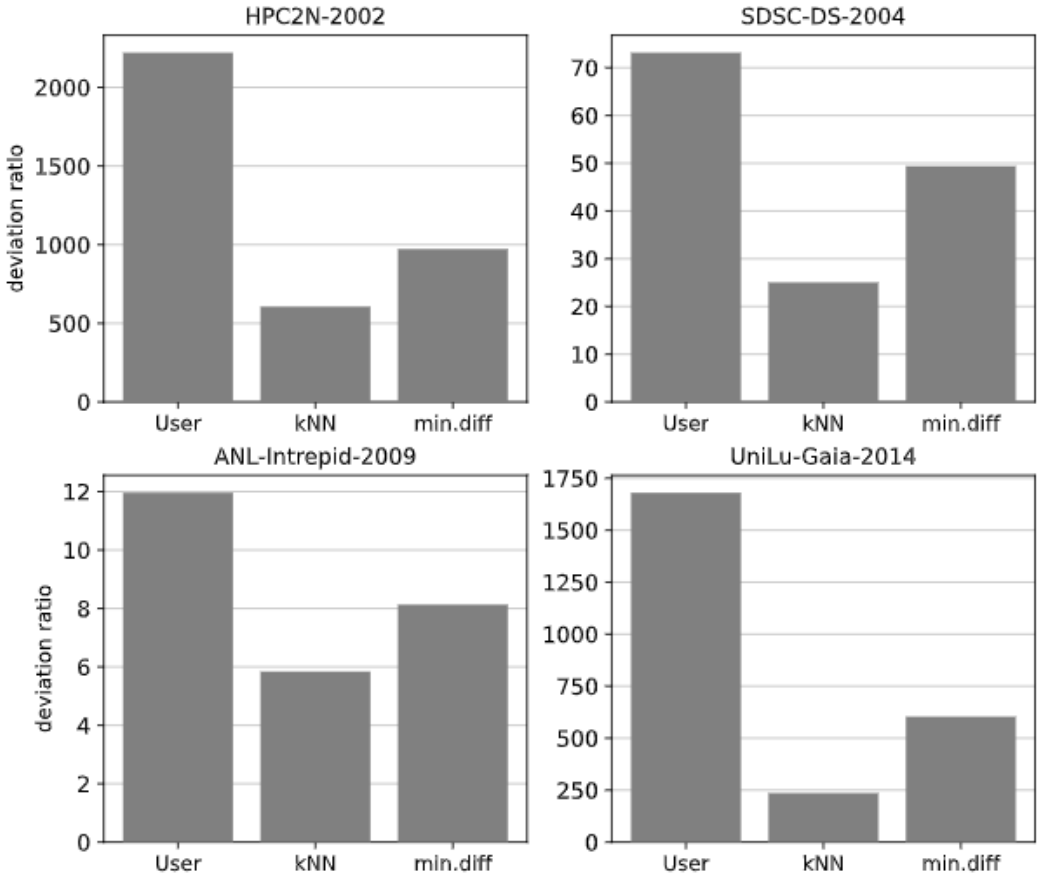
# Evaluation

- *k*NN intuitive configuration:
  - k = 5 most similar jobs
  - N = 1000 latest finished job
  - Θ = 0.001 (neighbor threshold)
  - ε = Θ/4
  - All attributes of the set A′ are normalized to the range [0,1].
- **Conservative Backfilling**

# Evaluation

## Prediction Accuracy

Average prediction deviation ratio between estimated and actual walltime:

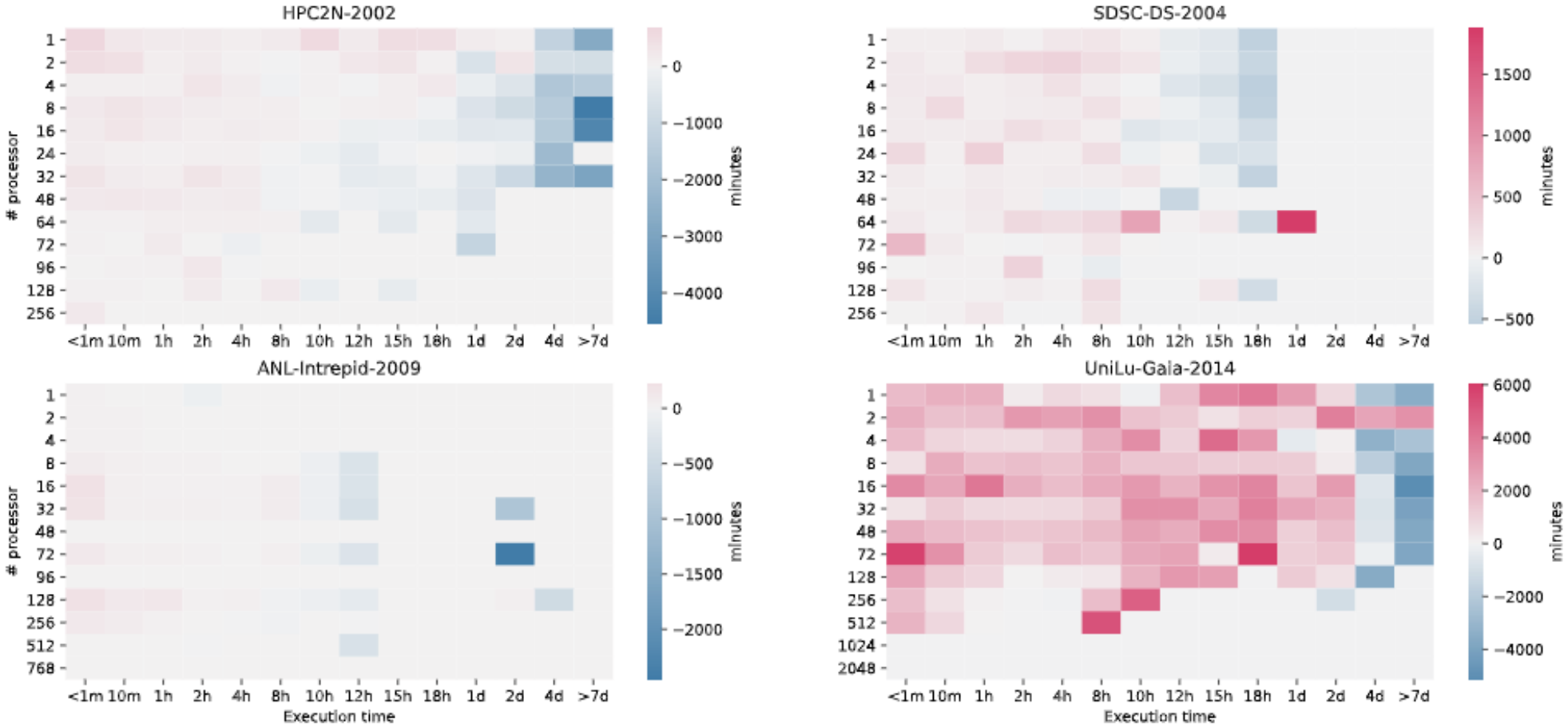



***k*NN has best accuracy**

# Evaluation

## Accuracy Improvement Heatmap

Prediction performance heatmap of *k*NN over original user estimates
**Red**: Improvement, **Blue**: Deterioration



# Evaluation

## Average Waiting Time

**Reflects the advantage of walltime improvement on each traces**

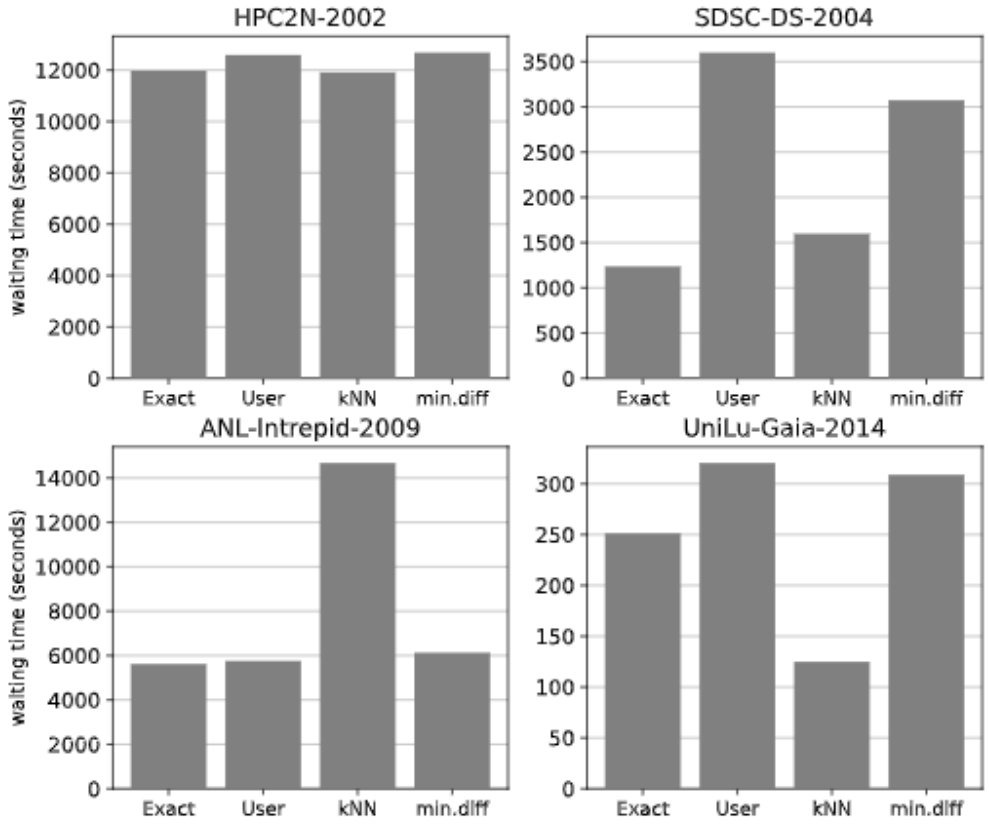
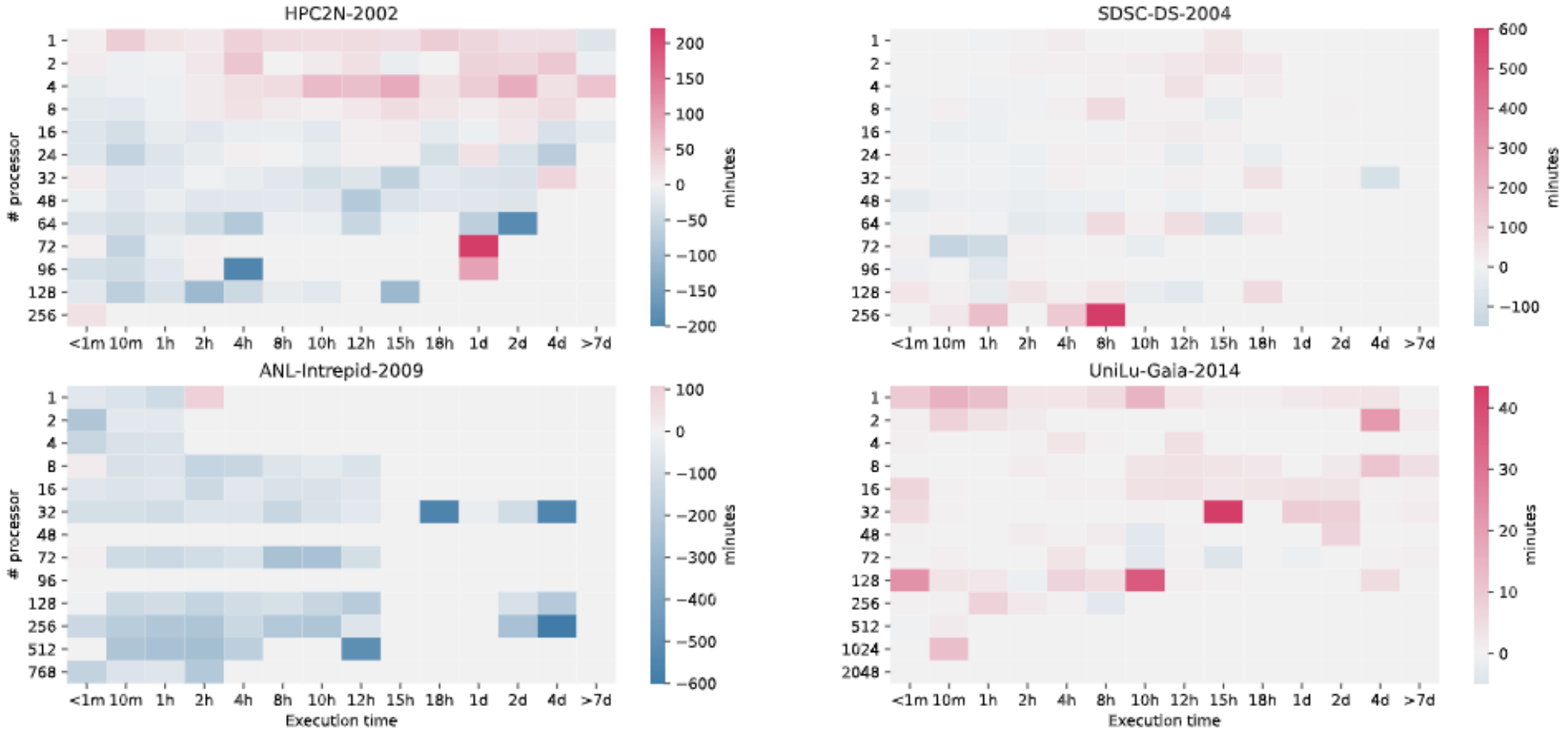



Figure 6. Comparision of average wait time

# Evaluation

## Waiting time Improvement Heatmap

**Wait time improvement or deterioration usingkNN on four workloads, comparing to user estimate**
**Red**: Improvement, **Blue**: Deterioration



# Conclusion and future work

- Soft walltime feature on OpenPBS has introduced an efficient and safe way to apply any advanced correction solutions such as *k*NN.
- The use of similar jobs to predict job walltime is good enough to deal with the complexity of workload and the uncertainty of user estimates.
- Even an intuitive setup can reduce the prediction deviation and thus improve system performance in some workloads

# Conclusion and future work

- There is still a challenge of picking better parameters for more reliable neighbors, especially in the case of large jobs
- The use of refined walltime must be taken carefully due to the gained benefits may not be as worth as expected

# Conclusion and future work

- Future work:
  - Deploy our *k*NN method on an HPC system running OpenPBS to fully examine the soft walltime in practice
  - Analyze other machine learning techniques with Soft Walltime to find a better approach for improving backfilling performance

# Thank you!

**Thanh Hoang Le Hai**

thanhhoang@hcmut.edu.vn
High Performance Computing Laboratory
HCMUT – VNU HCM