Performance and Energy Improvement of ECP Proxy App SW4lite under Various Workloads

Xingfu Wu, Valerie Taylor

Argonne National Laboratory and The University of Chicago

Zhiling Lan

Illinois Institute of Technology

SC2021 Workshop on Memory-Centric High Performance Computing

Nov. 14, 2021

Outline

- Background and Motivations
- ECP Proxy Application: SW4lite
- Performance Analysis
- Performance and Power Modeling using MuMMI and Ensemble Learning
- Performance and Energy Improvement
- Summary and Future Work

Background and Motivations

- Energy efficient execution of scientific applications requires insight into how HPC system features affect the performance and power of the applications.
- This insight generally results from significant experimental analysis and possibly the development of performance and power models.
- To balance power and performance for energy efficiency, one must understand the relationships between runtime, power, and the unique characteristics of a large-scale scientific application
- Insights about these relationships provide guidance for application optimizations to reduce runtime and power.
  - application code modification
  - system tuning
  - or a combination of both

ECP Proxy Application: SW4lite

- A bare bone version of SW4 (Seismic Waves, 4th order accuracy), which implements substantial capabilities for 3-D seismic modeling and uses a fourth order in space and time finite difference discretization of the elastic wave equations in displacement formulation

- It has the hybrid MPI/OpenMP implementations with C or Fortran and CUDA-aware MPI implementation

- It consists of five main kernels within a time step loop:
  – BC comm: communication among MPI tasks for exchanging boundary conditions (BC)
  – BC phys: imposing physical boundary conditions
  – Scheme: evaluating the difference scheme for divergence of the stress tensor
  – Supergrid: evaluating supergrid damping terms
  – Forcing: evaluating the forcing functions

ECP Proxy Application: SW4lite

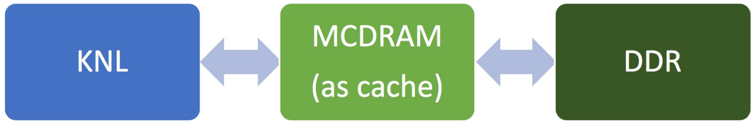
- Representative of SW4 with respect to computation and memory behavior and similar communication patterns
- SW4lite version 1.1 (the latest) provides five different types of input problems:
  - LOH.1, LOH.2, topo, cartesian, and pointsource
  - Strong scaling
- Often used to evaluate and compare different hardware architecture solutions for performance testing
- Distributed as one of ECP proxy applications
- Performance optimization of SW4lite itself has not been considered seriously

Contributions of This Paper

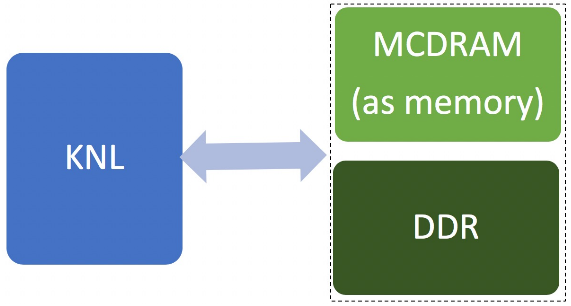
- We investigate performance and energy of SW4lite
  - under various workloads
  - with two different memory modes
  - on Cray XC40 Theta at Argonne
- We utilize hardware performance counter-based performance and power modeling to identify performance and power bottlenecks
  - Based on the insights from the performance and power models
  - To provide the most important counters which impact performance and power
- We use performance counter-guided optimization strategies to improve the performance and energy of SW4lite with the focus on memory-centric optimization and code modifications to achieve
  - up to 26.97% performance improvement
  - up to 19.44% energy saving on up to 16,384 cores

Cray XC40 Theta at Argonne

System Name	Cray XC40 Theta
Architecture	Intel KNL
Number of nodes	4392
CPU cores per node	64
Sockets per node	1
CPU type and speed	Xeon Phi KNL 7230 1.30GHz
L1 cache per core	D:32KB/I:32KB
L2 cache per socket	32MB (shared)
L3 cache per socket	None
Memory per node	16GB MCDRAM/192GB DDR4
Threads per core	4
Network	Cray Aries Dragonfly
Power tools	CapMC/PoLiMEr
TDP per socket	215W
Power Management	Yes
File System	Lustre PFS



Cache Mode



Flat Mode

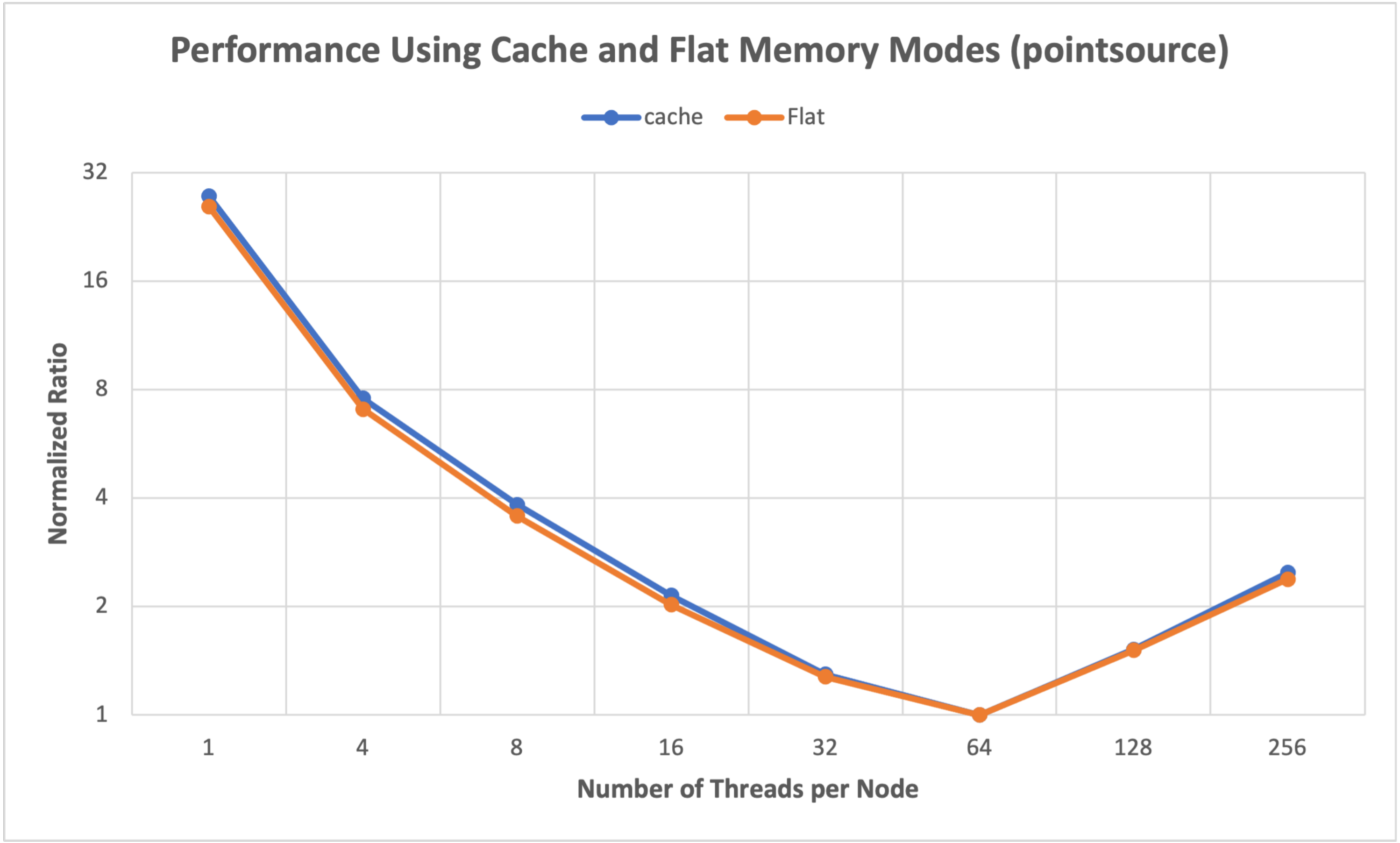
Performance Analysis Using Two Memory Modes

- Analyze performance of SW4lite using
  - Cache memory mode
  - Flat memory mode

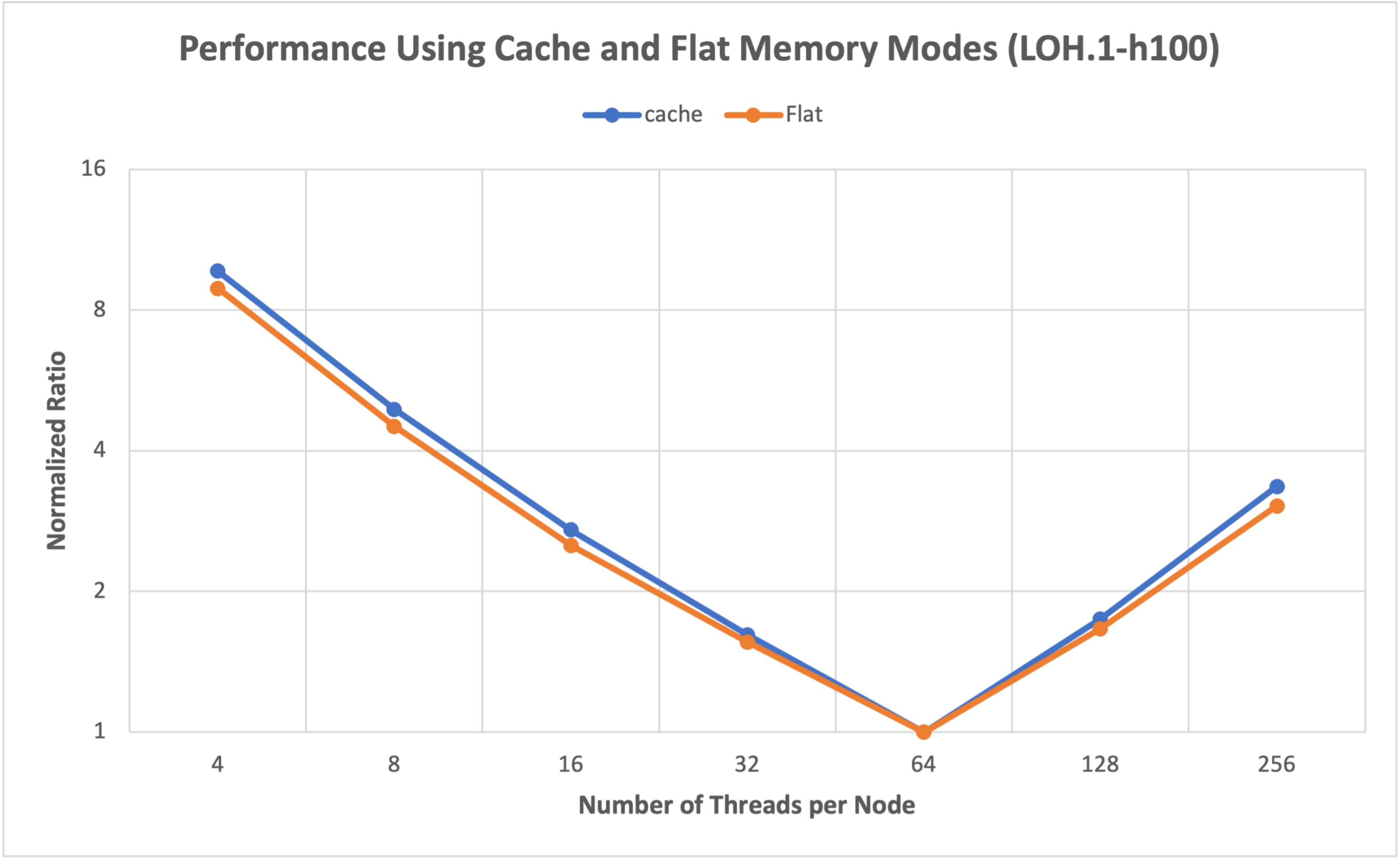
- Use two problems:
  - Small problem: pointsource
  - Large problem: LOH.1-h100

- Use up to 256 OpenMP threads per node on a single node

Performance Analysis (small problem)



Performance Analysis (large problem)



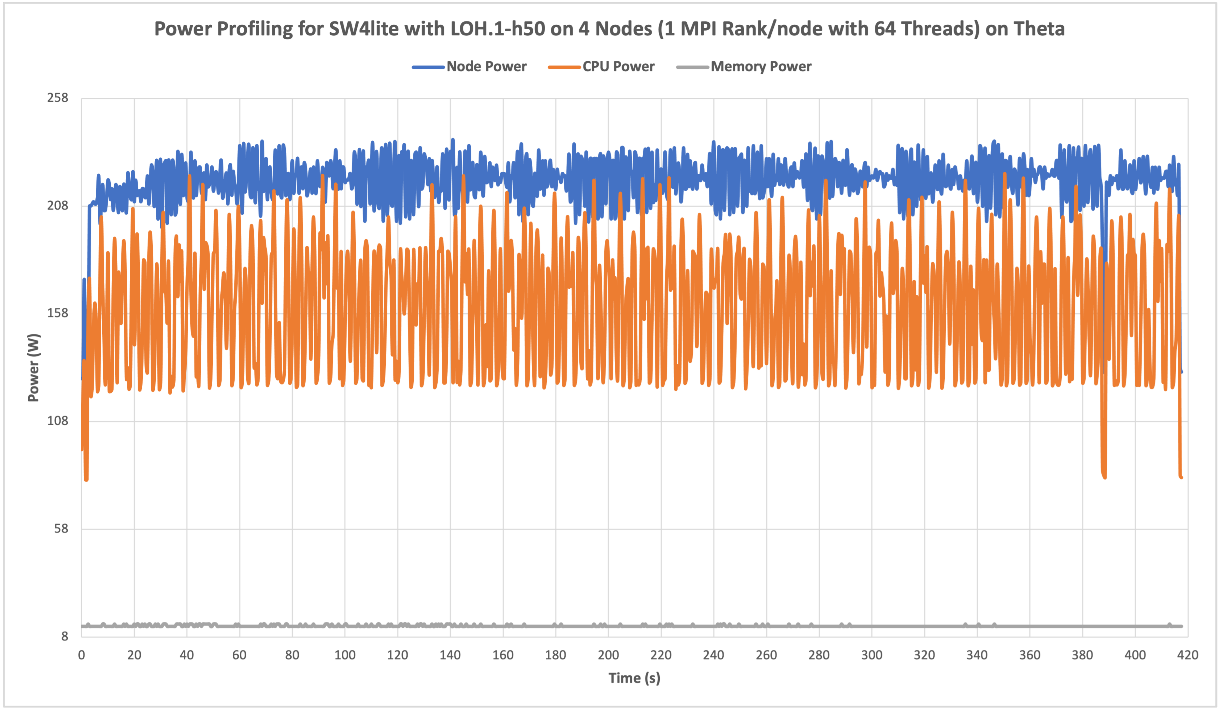
Performance Comparison using 64 threads per node

Problem	pointsource	LOH.1-h100
Flat/cache ratio	1.06	1.08

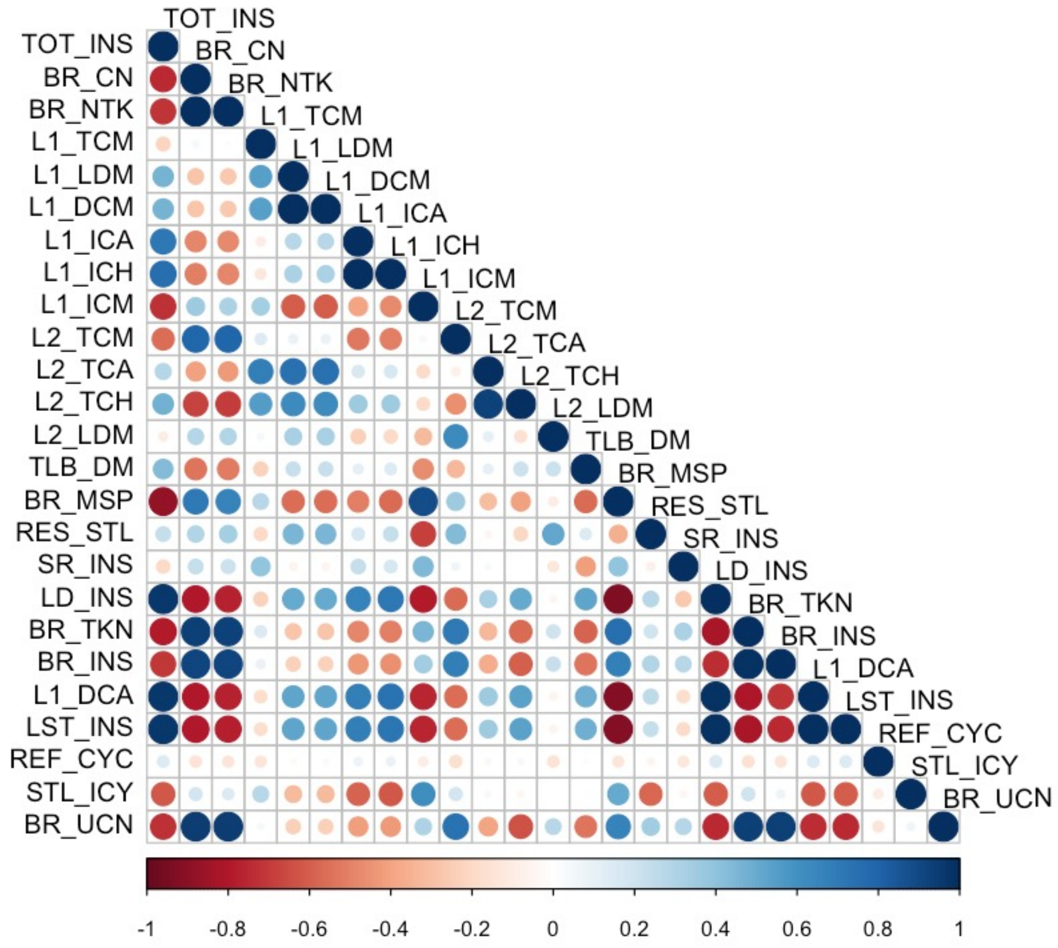
- Using 64 OpenMP threads per node with one thread per core results in the best performance for both cache and flat memory modes for SW4lite
- Using the cache mode outperforms using the flat mode
  - SW4lite with the problem sizes used fits into the high bandwidth MCDRAM
- Use 64 OpenMP threads per node and one thread per core with the cache mode for all of our experiments

Data Collection with Various Workloads

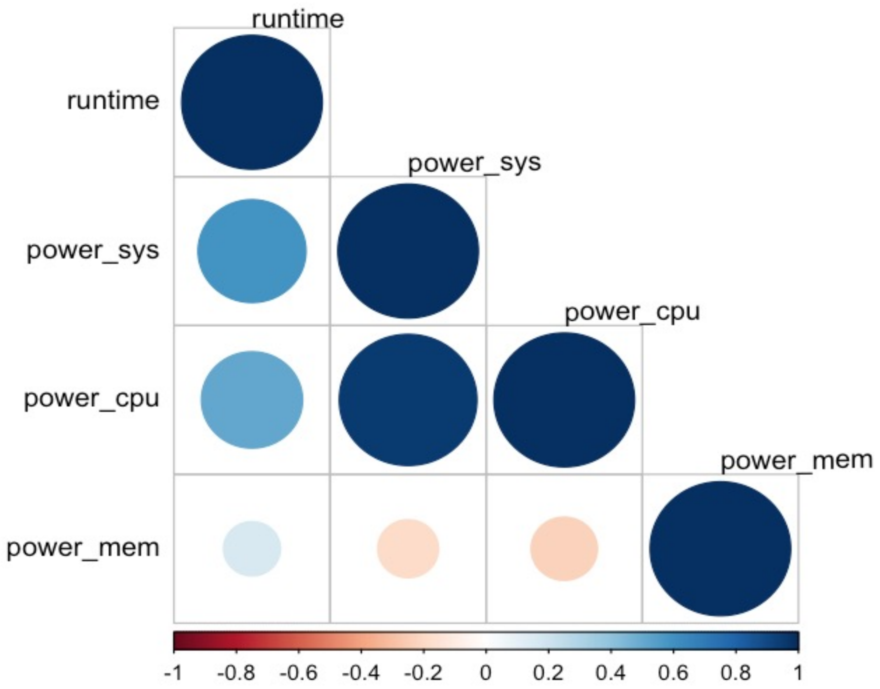
- Use the medium and large problems such as topo (gaussianHill.in, skinny.in), LOH.1 (LOH.1-h100.in, LOH.1-h50.in), and LOH.2 (LOH.2-h100.in, LOH.2-h50.in) to collect the data for 91 different configurations on up to 1024 nodes
- For each configuration, use MuMMI with PAPI to collect 26 available performance counters, and four metrics such as runtime, node power, CPU power, and memory power



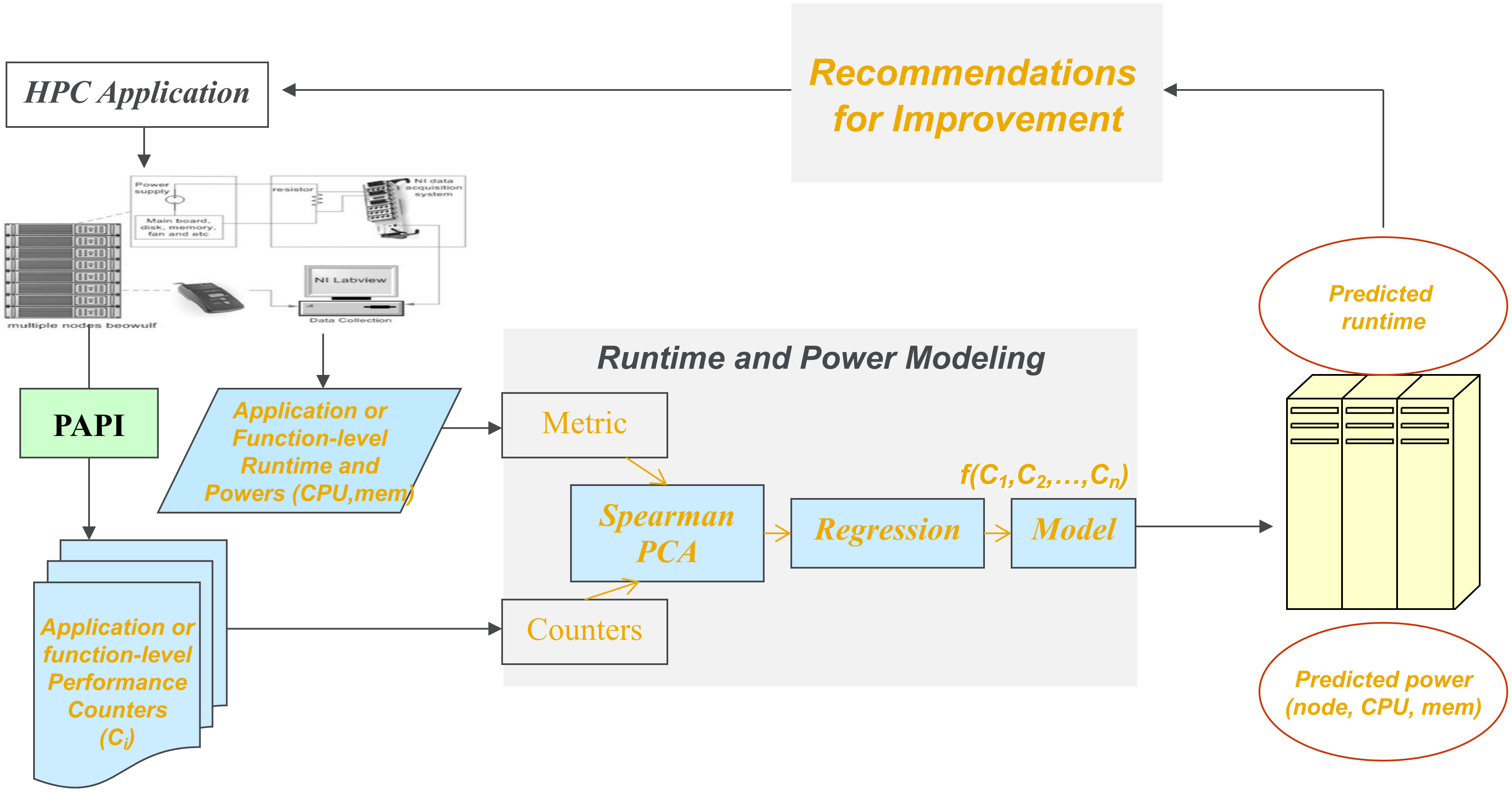
Data Collection



Data Collection



MuMMI: Counter-based Modeling Framework

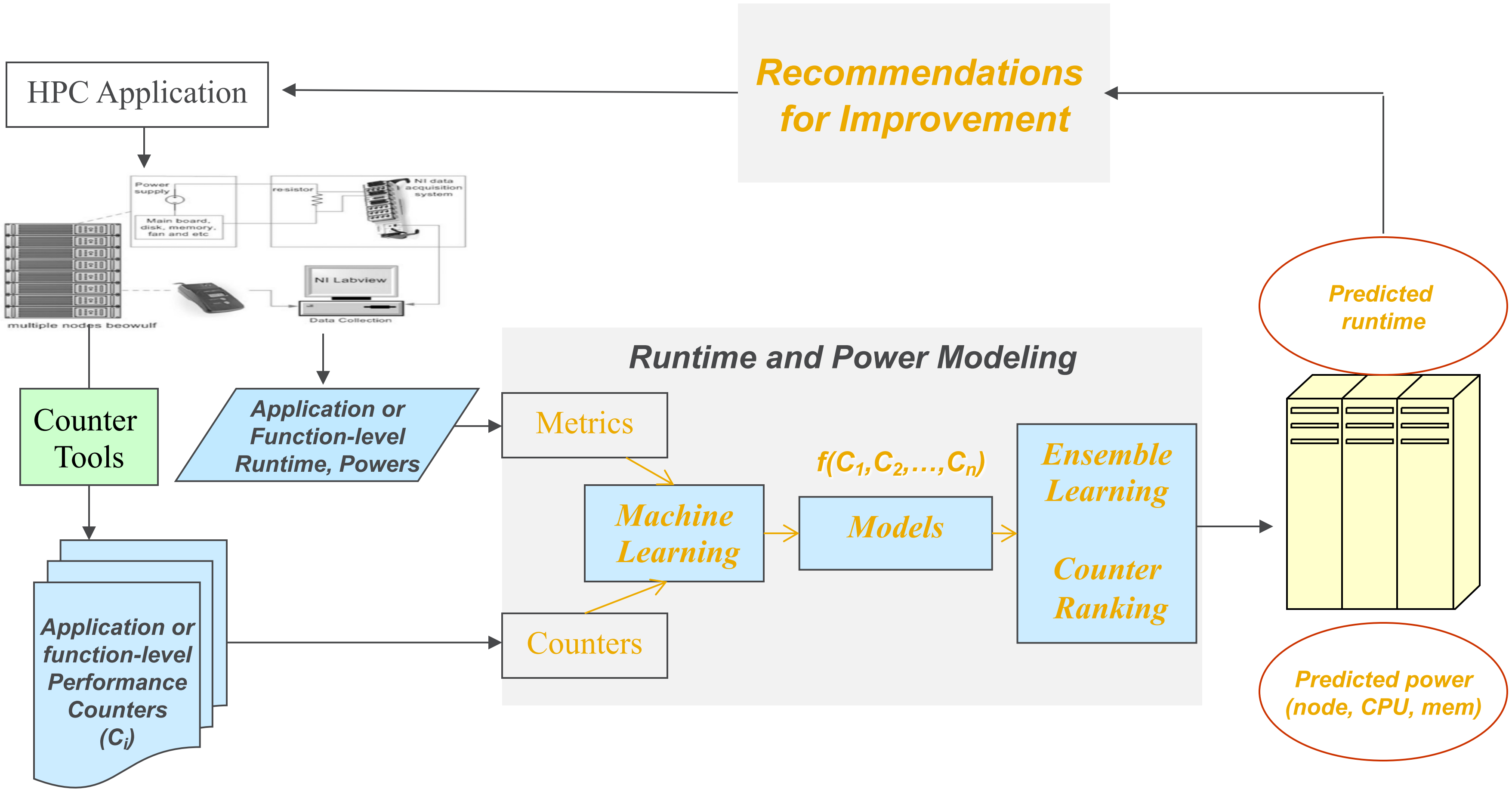


HPC Application
Recommendations for Improvement
Power supply
Main board, disk, memory, fan and etc
resistor
NI data acquisition system
NI Labview
multiple nodes beowulf
Data Collection
PAPI
Application or Function-level Runtime and Powers (CPU,mem)
Runtime and Power Modeling
Metric
Spearman PCA
Regression
$f(C_1,C_2,\ldots,C_n)$
Model
Counters
Application or function-level Performance Counters (C_i)
Predicted runtime
Predicted power (node, CPU, mem)

Four metrics: runtime, node power, CPU power, memory power

X. Wu, V. Taylor, J. Cook, and P. Mucci, "Using performance-power modeling to improve energy efficiency of HPC applications," IEEE Computer, vol. 49, no. 10, pp. 20–29, Oct. 2016.

MuMMI with Ensemble Learning



HPC Application
Recommendations for Improvement
Power supply
Main board, disk, memory, fan and etc
resistor
NI data acquisition system
NI Labview
multiple nodes beowulf
Data Collection
Predicted runtime
Runtime and Power Modeling
Counter Tools
Application or Function-level Runtime, Powers
Metrics
$f(C_1, C_2, \ldots, C_n)$
Ensemble Learning
Machine Learning
Models
Counter Ranking
Application or function-level Performance Counters (C_i)
Counters
Predicted power (node, CPU, mem)

Four metrics: runtime, node power, CPU power, memory power

X. Wu and V. Taylor, "Utilizing ensemble learning for performance and power modeling and improvement of parallel cancer deep learning CANDLE benchmarks," Concurrency and Computation: Practice and Experience, vol. e6516, 2021.

Performance and Power Modeling

- Apply MuMMI and ensemble learning to the dataset to build performance and power models based on performance counters
- Identify the most important counters from each model
- Consider the most important counters which impact performance and power for energy efficiency
- Optimize the application and/or tune the system

Performance Counter Ranking

TOP 3 PERFORMANCE COUNTERS FOR EACH MODEL USING MUMMI

Models	Top 3 performance counters		
Runtime	REF_CYC	BR_INS	
Node Power	L2_TCH	TLB_DM	LD_INS
CPU Power	L2_TCH	LD_INS	TLB_DM
Memory Power	BR_MSP	BR_UCN	STL_ICY

TOP 3 PERFORMANCE COUNTERS FOR EACH MODEL USING MVTBOOST

Models	Top 3 performance counters		
Runtime	L1_ICM	SR_INS	L1_TCM
Node Power	BR_CN	BR_UCN	LD_INS
CPU Power	LD_INS	BR_UCN	L1_ICM
Memory Power	REF_CYC	BR_UCN	TLB_DM

TOP 3 PERFORMANCE COUNTERS FOR EACH MODEL USING RFE

Models	Top 3 performance counters		
Runtime	BF_MSP	L1_ICM	BR_NTK
Node Power	BR_CN	BR_UCN	BR_NTK
CPU Power	BR_TKN	LD_INS	L1_ICM
Memory Power	TLB_DM	L1_DCA	REF_CYC

Performance Counter Ranking

- Using MuMMI, the dominant performance counters are
  - REF_CYC in runtime model
  - L2_TCH in node power and CPU power models
  - BR_MSP in memory power model
- Using mvtboost, the dominant performance counters are
  - L1_ICM in runtime model
  - BR_CN in node power model
  - LD_INS in CPU power model
  - REF_CYC in memory power model
- Using RFE, the dominant performance counters are
  - BR_MSP in runtime model
  - BR_CN in node power model
  - BR_TKN in CPU power model
  - TLB_DM in memory power model

Performance and Energy Improvement

- Our potential optimization efforts are threefold:
  - to address the dominance of REF_CYC in the runtime model using MuMMI and in the memory power model using mvtboost
    - indicates that something else other than CPU performance limits the performance and power of SW4lite.
  - to improve the cache utilizations (L1_ICM, L2_TCH)
  - to reduce TLB_DM

Looking into the SW4lite Source Code

- Focus on three main kernels for some hints:
  - Scheme entails evaluating the difference scheme for divergence of the stress tensor
  - Supergrid entails evaluating supergrid damping terms
  - Forcing entails evaluating the forcing functions
- Find that one issue is to apply ”#pragma omp for” to the for statement ”for (k = 1;k ≤ 6;k++)” in the five files: rhs4sg.C, rhs4sg_rec.C, rhs4sg_recNW.C, rhs4sgcurv.C and rhs4sgcurv_rev.C from SW4lite source code

```
#pragma omp for
for( k=1 ; k<= 6 ; k++ )
   for( j=jfirst+2; j<=jlast-2; j++ )
      #pragma ivdep
      for( i=ifirst+2; i<=ilast-2; i++ )
      {
         ...
      }
```

```
for( k=1 ; k<= 6 ; k++ )
   #pragma omp for
   for( j=jfirst+2; j<=jlast-2; j++ )
      #pragma ivdep
      for( i=ifirst+2; i<=ilast-2; i++ )
      {
         ...
      }
```

Number of Nodes	Original	Revised	Improvement (%)
1	96.38	78.71	18.33
2	54.19	46.30	14.56
4	36.99	31.51	14.81
8	22.97	20.22	11.97

Argonne NATIONAL LABORATORY

Looking into the SW4lite Source Code

- Further, to improve the cache utilization, we put ”#pragma unroll(6)” directive right before the loop to unroll the revised loop to improve the cache performance
- The ”#pragma unroll(6)” directive is a compiler optimization for loop unrolling to reduce loop control overhead

```
for( k=1 ; k<= 6 ; k++ )
   #pragma omp for
   for( j=jfirst+2; j<=jlast-2; j++ )
      #pragma ivdep
      for( i=ifirst+2; i<=ilast-2; i++ )
      {
         ...
      }
```

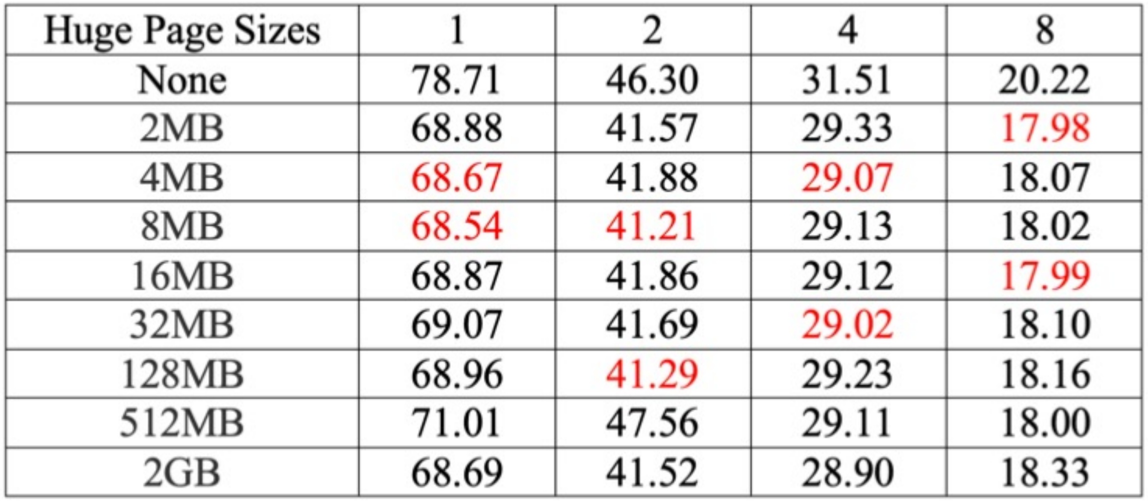
```
#pragma unroll(6)
for( k=1 ; k<= 6 ; k++ )
   #pragma omp for nowait
   for( j=jfirst+2; j<=jlast-2; j++ )
      #pragma ivdep
      for( i=ifirst+2; i<=ilast-2; i++ )
      {
         ...
      }
```

Number of Nodes	Revised	Revised + unrolling	Improvement (%)
1	78.71	74.83	4.93
2	46.30	44.71	3.43
4	31.51	31.40	0.35
8	20.22	18.32	9.40

Argonne NATIONAL LABORATORY

Utilizing Huge Pages

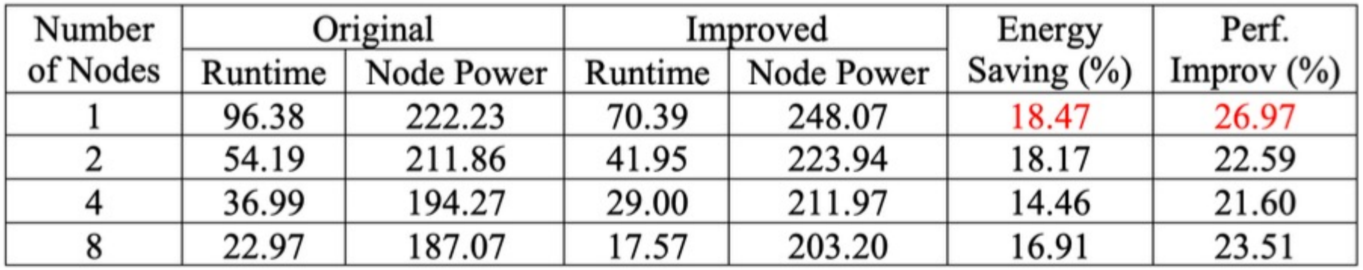
- TLB_DM occurs in several power models
- We can utilize huge page sizes to reduce TLB_DM so that the cache and memory performance may be improved
- Cray XC40 Theta provides the system modules for the huge page sizes from 2MB to 2GB.
  - The default page size is 4KB
- Observe that 8MB huge page resulted in the consistent better performance



Huge Page Sizes	1	2	4	8
None	78.71	46.30	31.51	20.22
2MB	68.88	41.57	29.33	17.98
4MB	68.67	41.88	29.07	18.07
8MB	68.54	41.21	29.13	18.02
16MB	68.87	41.86	29.12	17.99
32MB	69.07	41.69	29.02	18.10
128MB	68.96	41.29	29.23	18.16
512MB	71.01	47.56	29.11	18.00
2GB	68.69	41.52	28.90	18.33

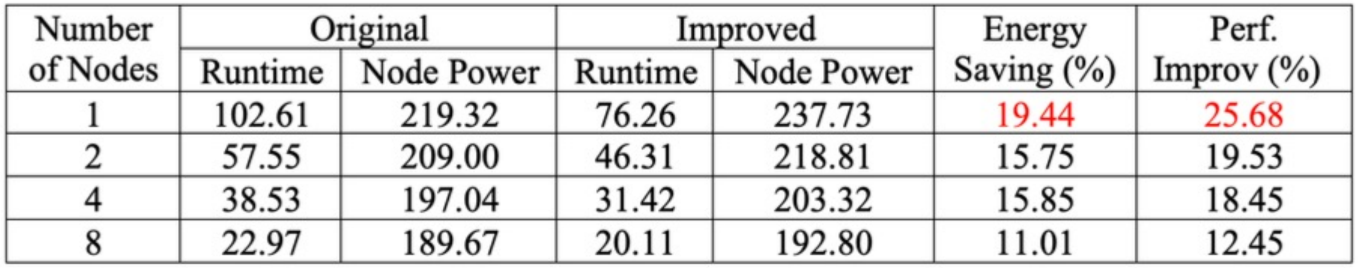
Combination of Improvement Strategies

- Apply the combination of the revised code with the loop unrolling and utilizing 8MB huge page (called the improved code) to the large problems
  - How much performance improvement and energy saving
  - On up to 16,384 cores (256 nodes) on Theta



Number of Nodes	Original		Improved		Energy Saving (%)	Perf. Improv (%)
	Runtime	Node Power	Runtime	Node Power		
1	96.38	222.23	70.39	248.07	18.47	26.97
2	54.19	211.86	41.95	223.94	18.17	22.59
4	36.99	194.27	29.00	211.97	14.46	21.60
8	22.97	187.07	17.57	203.20	16.91	23.51

LOH-1.h100



Number of Nodes	Original		Improved		Energy Saving (%)	Perf. Improv (%)
	Runtime	Node Power	Runtime	Node Power		
1	102.61	219.32	76.26	237.73	19.44	25.68
2	57.55	209.00	46.31	218.81	15.75	19.53
4	38.53	197.04	31.42	203.32	15.85	18.45
8	22.97	189.67	20.11	192.80	11.01	12.45

LOH-2.h100

Combination of Improvement Strategies

Number of Nodes	Original		Improved		Energy Saving (%)	Perf. Improv (%)
	Runtime	Node Power	Runtime	Node Power		
1	1620.11	241.67	1310.53	259.27	13.22	19.11
2	804.95	226.30	660.84	238.31	13.55	17.90
4	419.09	219.67	366.67	227.04	9.57	12.51
8	219.02	216.93	197.52	222.82	7.37	9.82
16	124.69	206.61	112.70	210.04	8.12	9.62
32	69.51	204.87	64.03	207.15	6.86	7.88
64	47.53	188.84	44.99	189.43	5.05	5.34
128	31.16	201.17	28.97	200.81	7.19	7.03
256	24.50	182.22	23.57	181.53	4.16	3.80

LOH-1.h50

Number of Nodes	Original		Improved		Energy Saving (%)	Perf. Improv (%)
	Runtime	Node Power	Runtime	Node Power		
1	2248.09	242.20	1901.94	244.93	14.44	15.40
2	1202.57	211.73	932.33	243.26	10.93	22.47
4	553.82	216.84	470.28	228.74	10.42	15.08
8	286.98	215.66	245.34	223.50	11.40	14.51
16	148.95	213.87	131.37	217.84	10.17	11.80
32	82.51	200.44	75.52	202.42	7.57	8.47
64	52.83	193.80	49.56	195.56	5.34	6.19
128	32.92	197.09	30.74	196.52	6.89	6.62
256	24.47	183.39	22.65	183.87	7.20	7.44

LOH-2.h50

Argonne NATIONAL LABORATORY

Summary

- We conducted the experiments to evaluate the performance of SW4lite with two memory modes (cache and flat) on Theta and found that using 64 OpenMP threads with the cache mode resulted in the best performance for SW4lite

- We applied MuMMI and ensemble learning to build the performance and power models to identify the most important performance counters for the potential optimization efforts

- we improved the performance and energy of SW4lite with the focus on the memory-centric application optimizations such as cache memory mode, loop unrolling and 8MB huge page and the source code modifications to achieve up to 26.97% performance improvement and up to 19.44% energy saving

- For the future work, we will apply our ytopt autotuning framework to further tune the performance and energy of SW4lite and other ECP proxy applications

Acknowledgements

- This work was supported in part by
  - DoE ECP PROTEAS-TUNE
  - DoE ASCR RAPIDS2
  - NSF grants CCF-1801856, CCF-2119203
  - Use of ALCF Theta under ALCF EE-ECP project