Postprint of Parallel and Optimization Methods for NSGA-II Based on Sunway Many-Core Processor

Authors: Liu Yao, Zheng Lin, Zheng Kai, Wang Su, Liao Qidan

Date: 2018-11-29T00:00:00+00:00

Abstract

Sunway TaihuLight, comprised of Sunway many-core processors, is currently China' s highest-performance supercomputer, providing a hardware platform for large-scale NSGA-II computations. Based on the hardware architecture characteristics, we design a hybrid parallel NSGA-II with an "island-master-slave-enhanced" approach. Building upon the master-slave model, inter-slave-core register communication is leveraged to enable sharing of local data storage among slave cores within a core group. By optimizing the workflow, parallelization of additional algorithmic modules on slave cores is achieved. Techniques including DMA transfer, vectorization, double buffering, and memory optimization are employed to significantly enhance the speedup ratio. Experimental results demonstrate that the optimized parallel NSGA-II exhibits favorable speedup and scalability on Sunway many-core processors.

Full Text

Preamble

Parallelization and Optimization of NSGA-II Based on Sunway Many-Core Processor

Liu Yao[1,2], Zheng Lin[1,2], Zheng Kai[1,2], Wang Su[1,2]†, Liao Qidan[1,2]
(1. School of Computer Science & Software Engineering, East China Normal University, Shanghai 200062, China;
2. State Key Laboratory of Mathematical Engineering & Advanced Computing, Wuxi, Jiangsu 214215, China)

Abstract: The Sunway TaihuLight supercomputer, composed of Sunway many-core processors, represents China' s highest-performance computing platform and provides a hardware foundation for large-scale NSGA-II problem solving.

Leveraging the architectural characteristics of the processor, this paper designs a hybrid parallel NSGA-II algorithm combining an "island model with enhanced master-slave" approach. Building upon the master-slave paradigm, register communication between slave cores enables shared access to local data memory within a core group. The algorithm workflow is optimized to parallelize additional modules on slave cores. DMA transmission, vectorization, double buffering, and storage optimization techniques are employed to significantly improve speedup. Experimental results demonstrate that the optimized parallel NSGA-II achieves excellent speedup and scalability on Sunway many-core processors.

Keywords: Sunway many-core processor; NSGA-II; parallel genetic algorithm; multi-objective; parallel optimization

0 Introduction

The Sunway TaihuLight supercomputer, the world' s first system to exceed 100 petaflops with a peak performance of 125.4 PFlops, is built upon China' s independently developed Sunway many-core processors. Recent years have witnessed growing research on Sunway processors and increasing parallelization efforts across various applications. Hong et al. parallelized the PETSc mathematical library, achieving a 16.4× speedup on slave cores. Meng et al. ported the OpenFOAM computational fluid dynamics package, addressing compatibility issues and sparse matrix computation overhead to attain 8.03× speedup per core group. Zhao et al. accelerated the HOG feature extraction algorithm for pedestrian detection, reaching 83× speedup on high-resolution images. Yao et al. ported the NAMD molecular dynamics software, achieving 20× performance improvement. However, research on parallelizing genetic algorithms on Sunway processors remains limited.

Genetic algorithms simulate natural evolution to search for optimal solutions and can be categorized as single-objective or multi-objective variants. For single-objective problems, Zhao et al. proposed a two-level parallelization that partitions populations across core groups (islands) while parallelizing fitness evaluation on slave cores, achieving up to 31.85× speedup under heavy computational load but suffering poor acceleration under light load. For multi-objective optimization, common algorithms include VEGA, MOGA, NPGA-II, and NSGA-II. NSGA-II employs non-dominated sorting to approximate the Pareto front and uses crowding distance to ensure good solution distribution. However, as problem scale and complexity grow, demands on solution quality and runtime increase substantially. Shen implemented NSGA-II parallelization on Sunway processors but only parallelized objective function computation, resulting in slave-core speedup barely exceeding 1 under light load. While these efforts demonstrate basic parallel genetic algorithms on Sunway processors, they lack systematic parallel optimization methodologies.

This paper designs a hybrid "island-enhanced master-slave" parallel NSGA-II algorithm tailored to Sunway' s architecture. MPI enables island-based par-

allelism across core groups, while Athread facilitates master-slave parallelism within groups. Register communication between slave cores enables shared local memory access. Workflow optimization parallelizes additional modules on slave cores, improving utilization. DMA, vectorization, double buffering, and storage optimization significantly boost parallel efficiency. The island model with migration strategy extends the algorithm to large-scale Sunway clusters, demonstrating excellent scalability and addressing the poor acceleration under light load.

1 Sunway TaihuLight System

The Sunway TaihuLight system employs a high-density elastic supernode architecture with high-throughput composite networking and a multi-objective optimization-oriented efficient system design. The system comprises 40,960 SW26010 heterogeneous many-core processors, scaled through computer plugin boards, supernodes, and cabinets to form a high-speed computing system.

The SW26010 processor combines on-chip computing arrays with distributed shared memory in a heterogeneous many-core architecture. Each chip integrates four core groups totaling 260 computing cores. Each core group contains one Management Processing Element (MPE, master core) and one Computing Processing Element (CPE) array. Register-level data communication, multi-mode asynchronous data streaming, and fast synchronization mechanisms enhance collaborative efficiency. Each processor has 32 GB of memory (8 GB per core group). CPEs can directly access main memory or use DMA for batch transfers. CPEs in the same row or column can communicate via registers, with each CPE having 64 KB Local Data Memory (LDM) and 16 KB instruction memory.

2 Parallel NSGA-II Design

2.1 Parallel Architecture

Based on the Sunway architecture, we designed a hybrid "island-enhanced master-slave" parallel NSGA-II algorithm. As shown in [Figure 2: see original paper], subpopulations p1, p2, p3, and p4 represent four islands that evolve in parallel with periodic migration. Each subpopulation is further divided among slave cores within a core group, following a master-slave model. During computation, slave cores exchange data according to specific rules, forming an enhanced master-slave pattern. Within each island, initialization, selection, crossover, and mutation execute on the master core, while slave cores handle objective function evaluation, non-dominated sorting, and crowding distance calculation, fully exploiting their computational power. This achieves two-level hybrid parallelism for NSGA-II.

2.2 Single Core Group Parallelization

A single core group corresponds to one island using the "enhanced master-slave" model. Since objective function parallelization performance depends on computational workload and data volume, light loads with small data sizes often yield low acceleration. Therefore, we parallelize additional NSGA-II modules. Non-dominated sorting and crowding distance calculation constitute significant portions of total runtime, making them prime targets for parallel acceleration.

The Athread library facilitates master-slave acceleration programming by enabling convenient control and scheduling of threads within core groups to maximize computational performance. The parallel NSGA-II workflow is illustrated in [Figure 3: see original paper]. NSGA-II differs from single-objective genetic algorithms primarily through its non-dominated sorting before selection, followed by crowding distance calculation. The selection, crossover, and mutation operators remain similar to single-objective variants.

During objective function parallelization, individuals are independent and can be evenly distributed across slave cores. The process initializes and creates a thread group activating 64 slave cores, then loads individual data via DMA. While slaves compute, the master core waits; results are written back via DMA upon completion.

For non-dominated sorting, each individual must compare against all unassigned individuals, requiring global population information. With only 64 KB LDM per slave, storing all individuals is impossible. Direct main memory access incurs high communication overhead and poor parallel efficiency, necessitating further optimization.

Crowding distance calculation builds upon sorting results. Traditionally, the master sorts objective values per layer, sends them to slaves for distance computation, then aggregates results. This low parallelization degree requires improvement.

2.3 Parallel Optimization

Parallelization faces several challenges: (a) minimizing communication overhead for fixed-complexity objective functions; (b) non-dominated sorting and crowding distance calculation have high inter-individual dependency, requiring global data access despite slow main memory and limited LDM; (c) register communication has strict rules—send/receive counts must match, excessive communication fills buffers causing stalls, and simultaneous bidirectional sends create deadlocks; (d) while fast, register communication supports only 256-bit vectors per transfer, and excessive communication overhead degrades parallel efficiency. We address these through register communication, double buffering, vectorization, and storage optimization.

2.3.1 Register Communication As shown in [Figure 4: see original paper], SW26010' s CPE array supports row/column register communication with strict rules: (a) receive count must equal send count—excess receives cause stalls; (b) sends are non-blocking within buffer limits, but full buffers block further sends; (c) simultaneous bidirectional sends cause deadlock. We optimize non-dominated sorting and crowding distance accordingly.

For non-dominated sorting, we designed a Hamiltonian loop [Figure 5: see original paper] where each individual visits every other slave core exactly once per sorting layer. Adjacent cores communicate in fixed directions, forming a ring. This ensures deterministic communication patterns, avoiding unnecessary overhead and exceptions. During comparison, an individual compares with local individuals before passing to the next core. To determine completion, each core counts its unassigned individuals, propagating and accumulating the count around the loop. If sorting completes, a stop broadcast is issued; otherwise, the process continues.

For crowding distance, after sorting, each slave stores partial individuals. Instead of master-based sorting, we use the Hamiltonian loop to sort objective values directly on slaves. During the first pass, individuals compare with all others to count larger individuals in their layer, determining their rank upon returning. The second pass computes distances between adjacent individuals. Finally, all individuals return to their origin cores for normalization and accumulation.

2.3.2 Double Buffering For objective function evaluation, double buffering maximizes communication/computation overlap. We allocate twice the required data space in LDM to hold two data blocks simultaneously. While computing one block, the next block is transferred via DMA, and the previous block is written back. This hides latency effectively.

Double buffering performance depends on the communication/computation ratio. Multiple DMA calls incur overhead, and data must be 32 KB-aligned contiguous blocks. Therefore, each transfer should be as large as possible while maintaining alignment. Optimal transfer sizes must be determined experimentally for each task.

2.3.3 Vectorization SW26010's cores support 256-bit vector operations (e.g., four double-precision values simultaneously). Non-dominated sorting involves extensive fitness comparisons—vectorizing four double comparisons theoretically yields 4× speedup. Objective function evaluation benefits similarly from vectorized accumulation.

Alignment is critical: unaligned loads/stores trigger exceptions that the OS handles by splitting operations, severely degrading performance. For unaligned objective values, padding or careful partitioning is necessary.

2.3.4 Storage Optimization Individuals are typically stored as structures. Transferring entire structures wastes LDM and communication bandwidth on irrelevant data. Extracting only required fields creates non-contiguous storage, reducing cache efficiency and DMA performance. We reorganize data by storing decoded information contiguously before computation, then store fitness values contiguously for transfer back, minimizing overhead.

2.4 Multi-Core Group Parallelization

As problem scale increases, single core groups cannot accommodate large populations. We extend the algorithm to large clusters using MPI.

2.4.1 Island Model The island model partitions the initial population across core groups (one MPE + 64 CPEs per island). Each subpopulation evolves independently on its island [Figure 6: see original paper], with periodic migration to maintain diversity and prevent premature convergence. This model incurs minimal communication overhead, theoretically achieving near-linear speedup.

2.4.2 Migration Strategy To avoid local optima, islands exchange individuals every few generations. Each process sends individuals to n-1 other processes. Process m receives individuals based on its ID and current generation. Migrants are selected from the outermost non-dominated layers (best individuals), replacing lower-ranked individuals in the receiving island.

3 Experimental Results and Analysis

Experiments were conducted on SW26010 using ZDT1 and BinhKorn test functions with population size 64,000, 10 generations, crossover probability 0.8, mutation probability 0.1, and 64 slave cores by default.

ZDT1 function:
$f_1(x) = x_1$
$f_2(x) = g(x) \cdot [1 - \sqrt{x_1/g(x)}]$
$g(x) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$, where $x_i \in [0, 1], i = 1, 2, \ldots, n, n = 30$

BinhKorn function:
$f_1(x, y) = 4x^2 + 4y^2$, $0 \leq x \leq 5, 0 \leq y \leq 3$
$f_2(x, y) = (x - 5)^2 + (y - 5)^2$
Constraints:
$g_1(x, y) = (x - 5)^2 + y^2 \leq 25$
$g_2(x, y) = (x - 8)^2 + (y - 3)^2 \geq 7.7$

3.1 Solution Correctness

[Figure 7: see original paper] and [Figure 8: see original paper] compare Pareto fronts from serial and parallel NSGA-II for ZDT1 and BinhKorn. The solution

sets nearly coincide, verifying correctness, with parallel NSGA-II showing more uniform distribution.

3.2 Speedup and Scalability

[Figure 9: see original paper] shows slave-core speedup versus core group count (fixed total population). Both test functions exhibit similar trends: speedup increases then decreases, peaking at 2 core groups with 13.95× and 16.28× speedup respectively. and break down runtime and speedup for three modules: objective evaluation, non-dominated sorting, and crowding distance calculation. Non-dominated sorting achieves the highest speedup (exceeding 25×), while objective evaluation shows the lowest. Since sorting dominates total runtime, overall speedup follows its trend. BinhKorn' s constraints increase sorting' s runtime proportion, yielding higher overall speedup.

The limited LDM capacity restricts per-core workload. Small test functions have low computation relative to communication overhead, causing objective evaluation speedup below 1. This underscores the necessity of parallelizing sorting and crowding distance under light load. However, omitting objective parallelization would increase sorting communication overhead, so balancing both components remains necessary.

[Figure 10: see original paper] shows total speedup across multiple core groups. Speedup grows superlinearly, exceeding 40,000× on 64 core groups. This occurs because non-dominated sorting' s $O(N^2)$ complexity dominates runtime (2-3 orders of magnitude larger than other components). With fixed total population, increasing islands reduces per-island individuals, decreasing time superlinearly. This demonstrates excellent scalability for large-scale problems on Sunway.

3.3 Optimization Method Comparison

Using two core groups, [Figure 11: see original paper] quantifies optimization contributions. Register communication provides the most significant improvement (190.8% and 171.4% speedup increase over baseline parallel NSGA-II). Vectorization also proves effective (31.1% and 35.9% improvement). Double buffering and storage optimization contribute additional gains.

4 Conclusion

This paper designs and implements a hybrid "island-enhanced master-slave" parallel NSGA-II algorithm for Sunway many-core processors, integrating multiple optimization techniques. These design principles and methods can guide parallelization of other algorithms on Sunway systems. Key contributions include:

a) Enhanced master-slave parallelism using inter-core register communication to share local memory within core groups.

b) Workflow optimization enabling parallel execution of additional modules on slave cores, improving parallelization degree.

c) Comprehensive application of DMA, vectorization, double buffering, and storage optimization to significantly boost speedup, addressing poor acceleration under light load and ensuring efficiency across workloads.

d) Extension to large-scale clusters via island model and migration strategy, demonstrating excellent scalability.

Experiments verify correctness, achieving 13.95× and 16.28× slave-core speedup on ZDT1 and BinhKorn, with total speedup exceeding 40,000× on 64 core groups. However, limited LDM capacity remains challenging. Future work will focus on further memory access optimization to improve efficiency and cache hit rates.

References

[1] Yang Guangwen, Zhao Wenlai, Ding Nan, et al. "Sunway TaihuLight" and its application system [J]. Science, 2017 (3): 12-16.
[2] Liu Xin, Guo Heng, Sun Rujun, et al. The characteristic analysis and exascale scalability research of large scale parallel applications on sunway taihuLight supercomputer [J/OL]. Chinese Journal of Computers, 2018, 41 (24): 1-10.
[3] Hong Wenjie, Li Kenli, Quan Zhe, et al. PETSc' s heterogeneous parallel algorithm design and performance optimization on the sunway TaihuLight system [J]. Chinese Journal of Computers, 2017 (9): 2057-2069.
[4] Meng Delong, Wen Minhua, Wei Jianwen, et al. Porting and optimizing OpenFOAM on sunway TaihuLight system [J]. Computer Science, 2017 (10): 64-70.
[5] Zhao Meiting, Liu Yi, Liu Rui, et al. Acceleration of histogram of oriented gradient (HOG) based on sunway many-core processor [J]. Computer Engineering and Science, 2017 (4): 611-618.
[6] Yao Wenjun, Chen Junshi, Su Zhichao, et al. Porting and optimizing of NAMD on sunway TaihuLight system [J]. Computer Engineering and Science, 2017 (6): 1022-1030.
[7] Jiang Xuesong, Jiang Shugang, Zhang Yu, et al. Parallel FDTD computation of million cores with the domestically-made many-core supercomputer [J]. Journal of Xidian University: Natural Science, 2017 (6): 65-69, 128.
[8] He Wangquan, Liu Yong, Fang Yanfei, et al. Design and implementation of parallel C programming language for domestic heterogeneous many-core systems [J]. Journal of Software, 2017 (4): 764-785.
[9] Ao Yulong. Research on key optimizations of sparse matrix and stencil computation for the domestic large many-core system [D]. Beijing: University of Chinese Academy of Sciences, 2017.
[10] Sara K, Mohammad T. An improved parallel genetic algorithm for optimal sensor placement of wireless sensor networks [C]// Proc of ACM Conference on

Wireless Network Security. Switzerland: [s. n.], 2014: 261-
[11] Sanhueza C, Jiménez F, Berretta R, et al. PasMoQAP: a parallel asynchronous memetic algorithm for solving the multi-objective quadratic assignment problem [C]// Proc of Evolutionary Computation. Spain: IEEE Press, 2017: 1103-1110.
[12] Potuzak T. Distributed//parallel genetic algorithm for road traffic network division using a hybrid island model//step parallelization approach [C]// Proc of IEEE//ACM International Symposium on Distributed Simulation and Real Time Applications. London: IEEE Press, 2016: 170-177.
[13] Kseniya N, Alexey R. A parallel genetic algorithm approach for monitoring devices placement [C]// Proc of International Multi-Conference on Engineering, Computer and Information Sciences. 2017: 186-189.
[14] Ortega G, Filatovas E, Garzón E M, et al. Non-dominated sorting procedure for Pareto dominance ranking on multicore CPU and//or GPU [J]. Journal of Global Optimization, 2017, 69: 1-21.
[15] Su Zhaopin, Zhang Guofu, Jiang Jianguo, et al. Multi-objective approach to emergency resource allocation using none-dominated sorting based differential evolution [J]. Acta Automatica Sinica, 2017 (2): 195-214.
[16] Chen Zhiwang, Huang Xingwang, Chen Zhixing, et al. Non-dominated sorting cloud model algorithm for interval multi-objective optimization [J]. Computer Engineering and Applications, 2016, 53 (22): 143-149.
[17] Deng Kaiwen, Han Xiaoqing, Liang Chen. Optimal configuration of energy storage based on elitist non-dominated sorting genetic algorithm with improved multi-objective particle swarm optimization [J]. Science Technology and Engineering, 2017 (20): 171-177.
[18] Zhao Ruixiang, Zheng Kai, Liu Yao, et al. Hybrid parallel genetic algorithm based on sunway many-core processors [J]. Journal of Computer Applications, 2017 (9): 2518-2523.
[19] Zhang Wenqiang, Fujimura Shigeru. Multiobjective process planning and scheduling using improved vector evaluated genetic algorithm with archive [J]. IEEJ Transactions on Electrical and Electronic Engineering, 2012 (3):
[20] Zhang Rui, Chiong R. Solving the energy-efficient Job-Shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption [J]. Journal of Cleaner Production, 2016, 112 (1): 3361-3375.
[21] Mousa A A, Elattar E E. Best compromise alternative to EELD problem using hybrid multiobjective quantum genetic algorithm [J]. Applied Mathematics & Information Sciences, 2014, 8 (6): 2889-2902.
[22] Deb K, Agrawal S, Pratap A, et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II [C]// Proc of Parallel Problem Solving From Nature Vi Conference. Berlin: Springer, 2000: 849-858.
[23] Sheng Wanxing, Liu Keyan, Liu Yuan, et al. Optimal placement and sizing of distributed generation via an improved nondominated sorting genetic algorithm II [J]. IEEE Trans on Power Delivery, 2015, 30 (2): 569-578.
[24] Shen Huanxue. Parallel research of NSGA-II and implement on domestic many-core processor [D]. Shanghai: East China Normal University, 2018.

[25] Deb K, Thiele L, Laumanns M, et al. Scalable test problems for evolutionary multiobjective optimization [M]. London: Springer, 2005: 105-145.

Note: Figure translations are in progress. See original paper for figures.