RAMSES-HR5 시뮬레이션 코드 핫스팟 분석 및 성능 최적화 연구

(A Study on Hotspot Analysis and Performance Optimization of the RAMSES-HR5 Simulation Code)

정현미*, 이현조**, 정기문*, 채철주**

(Hyun Mi Jung, Hyunjo Lee, Kimoon Jeong, Cheol-Joo Chae)

요 약

본 연구는 천체물리학 분야에서 광범위하게 활용되는 RAMSES-HR5 시뮬레이션 코드의 성능 병목 현상을 체계적으로 분석하고, 이를 개선하기 위한 최적화 전략을 제안한다. RAMSES-HR5는 AMR 기반의 하이브리드 병렬 구조의 수치 시뮬레이션 코드이다. 그러나 HPC 환경에서 RAMSES-HR5는 스레드 간 부하 불균형, MPI 통신 병목, 반복적인 I/O 연산 등으로 인해 연산 자원의 활용 효율이 저하되고 전체 실행 시간이 지연되는 문제가 발생한다. 이에 본 연구에서는 Intel VTune Profiler를 이용하여 RAMSES-HR5 코드의 핵심 연산 모듈에 대한 프로파일링을 수행하고, CPU 사용률, 메모리 접근 패턴, MPI 통신 대기 시간을 기반으로 핫스팟 분석하였다. 주요 핫스팟을 개선하기 위해 OpenMP parallel loop 구조를 고정 분할에서 동적 스케줄링 방식으로 전환하고, MPI_Waitsome 및 MPI_Reduce+Bcast를 활용하여 통신 병렬성을 확보하였다. 또한 메모리 모니터링 및 출력 루틴의 호출 빈도를 줄여 불필요한 통신과 I/O 오버헤드를 감소시켰다. 최적화된 코드는 8코어 클라우드 환경에서 기존 대비 약 10~15%의 전체 실행 시간 단축 효과를 보였으며, 연산-통신 병렬 중첩, 스레드 부하 분산, 통신량 감소 측면에서 병렬 효율성이 향상되었음을 확인하였다.

■ 중심어 : RAMSES-HR5 ; 고성능 컴퓨팅 ; 병목 분석 ; 병렬 최적화 ; 우주 시뮬레이션

Abstract

This study systematically analyzes the performance bottlenecks of the RAMSES-HR5 simulation code, which is widely used in the field of astrophysics, and proposes optimization strategies to improve its computational efficiency. RAMSES-HR5 is a numerical simulation code capable of accurately modeling galaxy formation and evolution, based on a hybrid parallel structure that incorporates AMR. However, in HPC environments, RAMSES-HR5 experiences issues such as thread-level load imbalance, MPI communication bottlenecks, and frequent I/O operations, which degrade resource utilization and increase overall execution time. To address these issues, we conducted profiling of the core computational modules in RAMSES-HR5 using Intel VTune Profiler, analyzing CPU usage, memory access patterns, and MPI communication wait times to identify performance hotspots. To mitigate the identified hotspots, we replaced the static scheduling in OpenMP parallel loops with dynamic scheduling and introduced MPI_Waitsome and MPI_Reduce+Bcast to enhance communication concurrency. In addition, we reduced unnecessary communication and I/O overhead by lowering the frequency of memory monitoring and output routines. The optimized code demonstrated a 10 - 15% reduction in total execution time in an 8-core cloud environment, with improved parallel efficiency in terms of computation - communication overlap, thread workload balancing, and reduced MPI traffic.

■ keywords : RAMSES-HR5 ; HPC, Bottleneck Analysis ; Parallel Optimization ; Cosmological Simulation

Ⅰ. 서 론

HPC 환경에서 프로세서 코어 수와 연산 능력이

* 정회원, 한국과학기술정보연구원 슈퍼컴퓨팅기술개발센터
** 정회원, 한국농수산대학교 교양학부
이 논문은 2025년도 한국과학기술정보연구원(KISTI)의 기본사업으로 수행된 연구입니다.(과제번호:K25L1M2C2)

접수일자 : 2025년 10월 31일 게재확정일 : 2025년 11월 11일
교신저자 : 채철주 e-mail : chae.cheoljoo@gmail.com

급격히 증가함에 따라 대규모 과학 계산 코드를 엑사스케일 컴퓨팅 시대에 맞게 효율화하는 것이 중요해지고 있다[1,2]. RAMSES-HR5는 천체물리 시뮬레이션 분야에서 널리 사용되는 대표적 병렬 적응격자 기법 코드이다[3-5]. 그러나 기존 RAMSES-HR5는 단순한 도메인 분할에 기반한 병렬화로 인해 다수의 프로세서를 활용할 경우 작업 부하 분배 및 메모리 사용상의 비효율이 발생하여 성능 확장에 한계가 있다. 그러므로 HPC 환경에서 RAMSES-HR5와 같은 대규모 시뮬레이션 코드의 핫스팟을 식별하고 성능을 향상시키는 연구가 필수적이다[6].

그러므로 본 연구에서는 Intel VTune Profiler를 이용하여 핫스팟 분석하고 RAMSES-HR5 코드 내 주요 핫스팟 구간을 식별하고, 해당 구간에 대한 효율적인 병렬화 및 최적화 기법을 적용함으로써 실제 실행 시간을 단축하는 데 있다. 구체적으로, 코드 수준에서 OpenMP를 활용한 쓰레드 병렬화 및 동적 작업 분할 기법을 도입하여 코어 간 부하 불균형을 완화하고, MPI 통신 패턴 최적화를 통해 통신 병목을 감소시키며, 불필요한 파일 I/O 연산을 제거하여 전체적인 부하를 경감시키고자 한다. 이를 통해 HPC 시스템 상에서 RAMSES-HR5의 성능을 개선하고 대규모 시뮬레이션을 효율적으로 수행할 수 있다. 본 논문의 구성은 다음과 같다. 2장에서는 RAMSES-HR5 시뮬레이션 코드의 구조와 기본 연산 특성을 분석하고, 3장에서는 Intel VTune Profiler를 활용한 성능 프로파일링 절차 및 핫스팟 분석 결과를 제시한다. 4장에서는 OpenMP 및 MPI를 이용한 최적화 기법의 적용 방법과 성능 측정 실험 결과를 설명하고 5장에서는 결론을 맺는다.

Ⅱ. RAMSES-HR5 시뮬레이션 코드 분석

RAMSES-HR5는 우주론적 수치 시뮬레이션으로, 기존 RAMSES 코드를 기반으로 한 하이브리드 병렬화 구조를 통해 대규모 우주 구조 형성과정을 정밀하게 모사할 수 있도록 설계되었다. RAMSES-HR5는 은하 형성과 진화를 연구하기 위한 시뮬레이션으로 은하의 물리적 특성 및 대규모 환경 의존성 분석이 가능함을 제시하였다. 이와 같이 RAMSES-HR5는 은하 진화의 환경 의존성, 항성형성 과정, 대규모 우주 구조의 진화 등 다양한 천체물리 현상을 해석하는 데 있어 핵심적인 수치 실험 기반을 제공하고 있다. 그림 1은 RAMSES-HR5의 시뮬레이션 코드 구조이며, 표 1은 주요 모듈을 보여주고 있다.

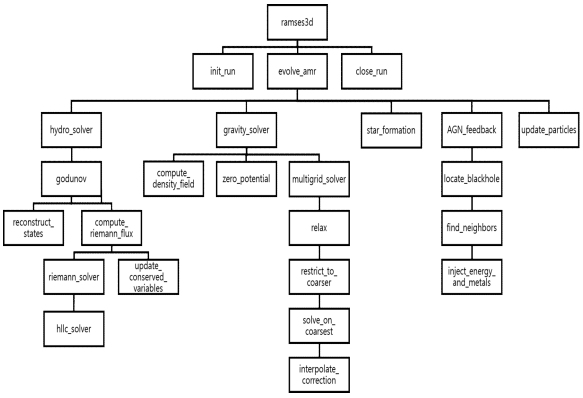

그림 1. RAMSES-HR5 시뮬레이션 코드 구조

표 1. RAMSES-HR5 주요 함수 개요

모듈명	목적	주요파일	주요함수	주요변수	라이브러리
AMR	격자 생성 및 관리	adaptive_loop, amr_commons	adaptive_loop, amr_refine, amr_coarsen	nlevelmax, ngridmax	MPI, OpenMP, HDF5
Hydro	유체 역학 해석	hydro_solver	hydro_solver, compute_fluxes	rho, pressure, velocity	HDF5, mathlib
Gravity	중력 해석	gravity_solver	gravity_solver, poisson_solver	potential, mass	FFT, MPI
Particles	입자 시뮬레이션	particles	particles_update, particles_accelerate	x, v, mass	MPI
IO	입출력 처리	amr_io	write_output, read_input	-	HDF5

Hydro 모듈은 천체 유체의 운동을 기술하는 Euler 방정식을 수치적으로 해석하는 모듈로 Godunov-type 방법 (Riemann Solver)을 사용하여 밀도(density, ρ), 속도 벡터(velocity, v), 압력(pressure, P), 에너지(energy, E) 물리량을 격자 기반으로 계산 및 갱신한다[7-8]. Hydro 모듈은 Riemann Solver를 통한 고정밀 해상도를 유지하고, MPI 병렬 처리 지원으로 대규모 시뮬레이션 가능하다는 장점이 있지만 Flux 계산 비용

이 높은 연산 집중 모듈이라는 단점이 있다. 표 2는 Hydro 모듈의 주요 기능을 보여주고 있다.

표 2. Hydro 모듈 주요 기능

기능	세부 내용
초기화 (init_hydro)	- 유체 변수 배열 (ρ, v, E) 메모리 할당 및 초기 조건 적용 - 유체 관련 상수 및 파라미터 초기화 (e.g., 격자 크기, 경계조건)
시간 적분 (hydro_solver)	- 주 시간 루프 내에서 Euler 방정식 해석 - Courant 조건에 따라 시간 스텝 계산 (CFL 조건) - 각 셀의 밀도, 속도, 압력 업데이트
Flux 계산 (compute_fluxes)	- Riemann Solver (HLLC, Roe 등) 호출 - 셀 경계에서 물리량 Flux 계산 (질량, 운동량, 에너지)
Shock 처리	- 충격파 해상도 보존 - 인공 점성 및 limiter 적용
냉각/가열 (cooling_fine.f90)	- Radiative Cooling, UV Background, Heating Process 반영
경계조건 처리 (boundana.f90)	- Domain Boundary에서 유체량 보존/반사/주입 등 처리
MPI 병렬화	- 각 CPU 도메인별 Flux 계산 후 Ghost Cell 정보 교환 - MPI 집계 연산으로 전역 물리량 계산 (e.g., 에너지 총합)

Gravity 모듈은 중력장 계산을 담당하며, Poisson 방정식을 풀어 격자/입자 기반의 중력 퍼텐셜과 가속도를 계산하고 중력 관련 계산을 위해 FFT(Fast Fourier Transform) 또는 Multigrid Solver를 사용한다[9]. HR5 버전에서는 글로벌 계산 오류를 제거하기 위해 FFT 사용한다. FFT 및 Multigrid Solver 병용 가능하기 때문에 다양한 경계조건 지원하다는 장점이 있지만 FFT 사용 시 주기적 경계조건 강제하고 Multigrid Solver 사용 시 성능 및 정확성 트레이드오프 존재한다는 단점이 있다. 표 3은 Gravity 모듈의 주요 기능을 보여주고 있다.

표 3. Gravity 모듈 주요 기능

기능	세부 내용
중력 퍼텐셜 계산 (gravity_solver)	- Poisson 방정식을 풀이하여 중력 퍼텐셜(φ) 계산
질량 밀도 분포 처리	- AMR 격자 및 입자 질량 분포(rho, mass)를 입력으로 사용
FFT/Multigrid Solver	- 선택적으로 FFT 기반 (주로 주기적 경계조건), 또는 Multigrid Solver 사용 (비주기적 경계 지원)
중력 가속도 계산	- 퍼텐셜의 기울기(∇φ)를 계산하여 각 셀/입자에 중력 가속도(acceleration) 할당
입자 중력 상호작용	- 입자-격자 상호작용: Particle-Mesh 방법 (CIC, TSC 등)
MPI 통신	- 중력 퍼텐셜 및 가속도의 경계 데이터 통신 처리 (Ghost Cell 교환)
Cosmology 지원	- 팽창 우주 모드(a(t), Hubble Flow) 처리 (aexp, H0 등)
Light-Cone 연계	- 관측자 Light-Cone 출력과 중력장 추출 연계

Particles 모듈은 N-body 시뮬레이션에서 입자의 동역학을 처리하는 핵심 모듈로 다양한 입자 기반 물리 현상을 처리한다[10]. Particle-Mesh 방법으로 대규모 N-body 시뮬레이션 처리 가능하며, AMR과의 통합으로 고해상도 지역에서 입자 분해능 증가한다는 장점이 있지만 Particle 수 증가에 따른 데이터 관리 비용 증가하고 Particle Load Imbalance으로 MPI Domain 분할 효율성을 저하한다는 단점이 있다. 표 4는 Partivles 모듈의 주요 기능을 보여주고 있다.

표 4. Particles 모듈 주요 기능

기능	세부 내용
입자 초기화 (particles_init)	- 초기 입자 위치(x), 속도(v), 질량(mass), ID 설정
입자 업데이트 (particles_update)	- 입자 위치 및 속도 갱신 (x = x + v*dt)
중력 가속도 반영 (particles_accelerate)	- 중력장(acceleration) 적용하여 속도 업데이트
입자-격자 상호작용	- Particle-Mesh (PM) 방법으로 격자-입자 질량 분포 매핑 (CIC, TSC Kernel 사용)
경계조건 처리	- Periodic / Reflective / Open Boundary 처리
MPI 병렬화	- 입자 분포에 따라 Domain Decomposition 후 Ghost Particle 교환
입자 출력	- 입자 데이터 (위치, 속도, 질량, ID 등) 저장
Light-Cone 연계	- light_cone.part.f90 호출하여 관측자 기준 입자 선택 및 출력

III. RAMSES-HR5 시뮬레이션 핫스팟 분석

본 연구에서는 구글 클라우드 환경에서 Intel® VTune Profiler v2025.4.0을 이용하여 코드 프로파일링을 수행하였다. Intel VTune Profiler는 CPU, Memory, Threading, MPI 통신 등 다양한 성능 병목 요소를 분석하는 도구로 대규모 시뮬레이션 코드의 병목 구간(AMR, Hydro, Gravity, Particles, I/O)을 분석하고, 최적화 방안을 도출할 수 있다. 표 5는 Intel VTune Profiler을 이용한 RAMSES-HR5 핫스팟 분석 방법을 보여주고 있다.

표 5. Intel VTune Profiler을 이용한 핫스팟 분석 방법

핫스팟 분석	- 명령어 : vtune -collect hotspots -r ./vtune_results -mpirun -np 64 ./ramses3d box_128.nml - collect hotspots: CPU 병목 분석 (기본 모드) - -r ./vtune_results: 결과 저장 디렉토리 - mpirun -np 64: MPI 64코어 실행 - ./ramses3d box_128.nml: RAMSES 실행 명령
MPI 분석	- 명령어 : vtune -collect mpi -r ./vtune_results_mpi -- mpirun -np 64 ./ramses3d box_128.nml - MPI 통신 시간, 동기화 오버헤드 등 분석
Memory Access 분석	- 명령어: vtune -collect memory-consumption -r ./vtune_results_mem - mpirun -np 64 ./ramses3dbox_128.nml - 메모리 병목 및 NUMA 효율성 분석
Roofline 분석	- 명령어 : vtune -collect roofline -r ./vtune_results_roofline - mpirun -np 64 ./ramses3d box_128.nml - FLOPS/Memory Bandwidth한계 분석

프로파일링 결과 OpenMP 병렬 영역 생성 및 동기화하는 kmp fork barrier & call 함수와 MPI 프로파일링 인터페이스 PMPI를 통해 제공되는 원래 MPI 함수의 warp 함수인 pmpi allreduce & waitall 함수에 대한 병목현상이 가장 심한 것으로 분석되었다. 그림 2는 핫스팟 분석 결과를 보여주고 있다.

Function	Module	CPU Time	% of CPU Time
__kmp_fork_barrier	libiomp5.so	6311.070s	41.1%
pmpi_allreduce_	libmpifort.so.12	1998.537s	13.0%
__kmp_fork_call	libiomp5.so	1706.718s	11.1%
pmpi_waitall_	libmpifort.so.12	1141.301s	7.4%
__intel_avx_rep_memset	ramses3d	417.669s	2.7%
[Others]	N/A*	3792.852s	24.7%

*N/A is applied to non-summable metrics.

그림 2. RAMSES-HR5 핫스팟 분석 결과

주요 핫스팟 함수 및 호출 지점을 분석하였을 때, MPI 관련 함수를 제외할 경우 그림 3과 같이 Hydro의 Godunov 함수의 비용이 가장 높았으며, 주요 작업 분석 결과 그림 4와 같이 AMR 색인 구조 조정에 많은 자원이 할당된다는 점을 확인할 수 있었다.

그림 3. 주요 핫스팟 함수 및 피호출 함수

그림 4. 주요 작업 분석 결과

OpenMP 병렬화에는 barrier가 필수적이나 동기화를 위해 모든 스레드가 끝날 때까지 대기해야하므로 임의의 스레드가 느리면 전체 속도가 저하된다. RAMSES-HR5의 AMR 구조에서는 어떤 스레드는 많은 leaf cell을 처리하는 반면, 어떤 스레드는 처리할 cell이 거의 없어 일찍 끝나고 대기한다는 점을 확인하였을 때, 작업량이 고르지 않아 불균형이 발생하여 이로 인해 barrier에서 대기 시간이 증가한다는 점을 확인하였다. 이를 개선하기 위해 본 연구에서는 OpenMP parallel do 구문에서 작업을 고정 분할하기 때문에 발생하는 문제를 동적인 방식인 !$omp parallel do schedule으로 작업을 실행 중에 동적으로 분할하는 방식으로 개선하고자 한다. 그림 5는 기존의 고정 분할 방식에서 동적 분할 방식으로의 코드 개선 부분을 보여주고 있다.

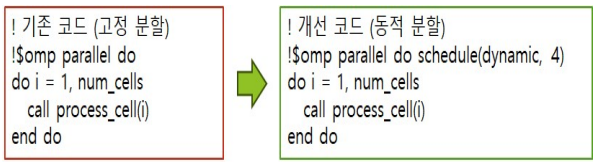

그림 5. 동적 분할 코드 개선 방안

Ⅳ. RAMSES-HR5 시뮬레이션 성능 최적화를 위한 코드 변환 및 성능 테스트

본 연구에서는 RAMSES-HR5에서의 주요 핫스팟 지점인 OpenMP 스레드 생성 및 종료 오버헤드, MPI 통신 대기, 전역 집단 통신 과부하를 개선하기 위해 OpenMP 구조는 유지하면서 불필요한 MPI 집단연산 및 동기화를 줄여 통신 부하를 완화하고 코드 구조는 안정성을 유지하면서 모니터링 비용만 최소화하는 방안을 제안한다. 메모리 사용량 모니터링 루틴(getmem, writemem)은 모든 coarse step마다 수행되며, 각 스텝마다 getmem()으로 메모리 측정, MPI_ALLREDUCE()로 전체 노드 메모리 합산, writemem()으로 루트 노드가 출력을 수행한다 [11-12]. 결과적으로 통신 및 I/O 오버헤드가 누적되어 주기적으로 계산이 지연되기 때문에 그림 6과 같이 adaptive_loop.F90 코드를 개선하였으며, ncontrol 변수를 도입하여 불필요한 통신과 파일 출력 빈도를 감소하였다.

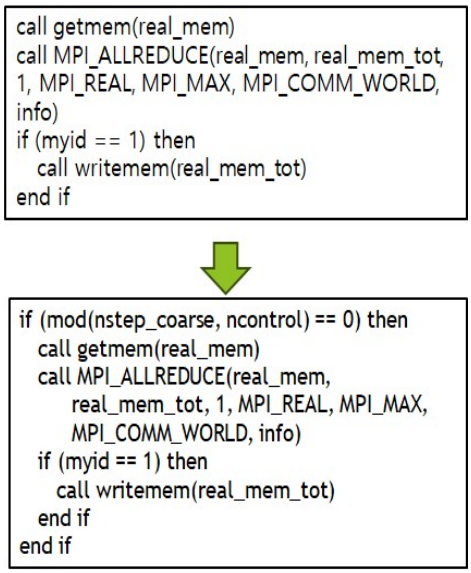

그림 6. adaptive_loop.F90 코드 개선 사항

RAMSES-HR5에서는 AMR 셀 간 경계 정보를 교환할 때 MPI_Waitall을 사용하여 모든 노드가 동시에 대기하고, 빠른 노드도 느린 노드가 끝날 때까지 기다리기 때문에 CPU idle 증가하며, 통신과 계산이 분리되어 중첩 불가능하다. 그러므로 그림 7과 같이 virtual_boundaries.F90 코드를 MPI_Waitall를 MPI_Waitsome으로 개선하여 도착한 이웃만 먼저 처리하고 나머지는 내부 계산 중첩하고 경계 타입별 (face/edge/corner) 파생 데이터형으로 패킹 효율을 개선하였다.

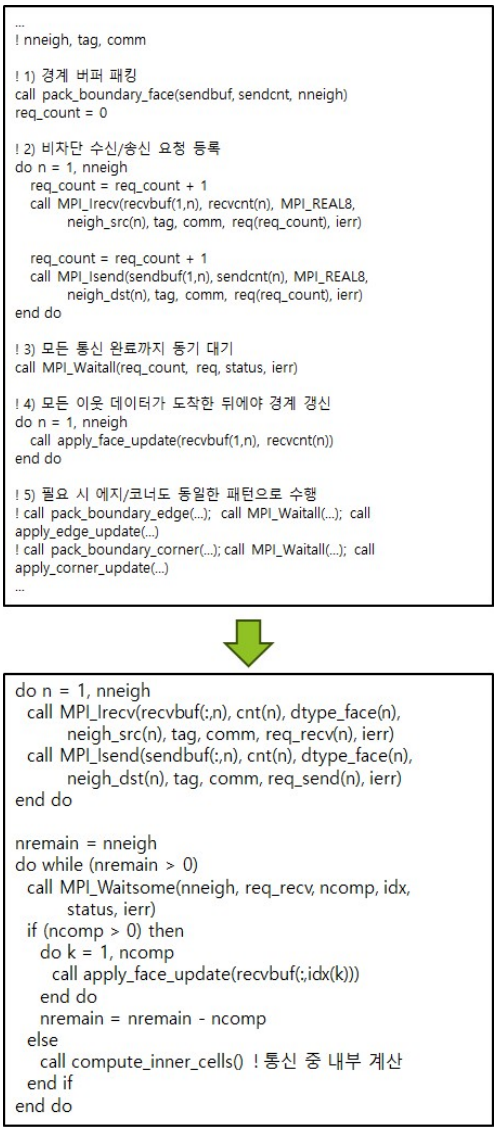

그림 7. virtual_boundaries.F90 코드 개선 사항

RAMSES-HR5에서는 부하 균형 시 각 노드의 min/max 부하를 MPI_Allreduce로 각각 계산하고

연속 2회 호출로 전역 동기화와 트래픽 과부하 발생하며, 노드 수가 증가할수록 비례하여 지연 증가한다. 그러므로 전역 통신 빈도를 축소하고 집단 연산을 계층적으로 수행하면서 불필요한 데이터 중복 송수신을 제거하기 위해 그림 8과 같이 load_balance.F90 코드를 MPI_Allreduce 2회에서 MPI_Reduce+MPI_Bcast 1회로 단축하고 intra-node와 inter-node를 분리하여 트래픽을 감소하였다.

```
...
integer :: ierr
real(8) :: local_min, local_max
real(8) :: global_min, global_max

! 각 프로세스의 지역 부하 계산 (셀 수, 연산량 등)
call compute_local_load(local_min, local_max)

! 모든 프로세스에서 최소 부하값 집계
call MPI_Allreduce(local_min, global_min, 1, MPI_REAL8, MPI_MIN,
MPI_COMM_WORLD, ierr)

! 모든 프로세스에서 최대 부하값 집계
call MPI_Allreduce(local_max, global_max, 1, MPI_REAL8, MPI_MAX,
MPI_COMM_WORLD, ierr)

! 루트 프로세스에서 결과 출력 또는 부하 재분배 판단
if (myid == 0) then
   write(*,*) 'Global load min = ', global_min
   write(*,*) 'Global load max = ', global_max
end if

! 부하 불균형 판단 및 분할 재계산
if (global_max / global_min > load_threshold) then
   call redistribute_load()
end if

...
```

↓

```
call MPI_Comm_split_type(comm_world,
        MPI_COMM_TYPE_SHARED, 0, MPI_INFO_NULL,
        shmcomm, ierr)
call MPI_Reduce(local_minmax, node_minmax, 2,
        MPI_REAL8, MPI_MIN, node_root, shmcomm, ierr)

if (rank_in_node == node_root) then
  call MPI_Reduce(node_minmax, global_minmax, 2,
        MPI_REAL8, MPI_MIN, world_root, intercomm, ierr)
end if

call MPI_Bcast(global_minmax, 2, MPI_REAL8, world_root,
        comm_world, ierr)
```

그림 8. load_balance.F90 코드 개선

RAMES-HR5 코드 개선 사항에 대해서 8코어 클라우드 환경에서 시뮬레이션한 결과 전체 실행 시간이 약 10~15% 단축되었음을 확인할 수 있었다. 이러한 실행 시간 단축은 adaptive_loop 코드에서 불필요한 동기화를 제거하고, virtual_boundaries 코드에서는 통신-계산을 동시에 수행하며, load_balance 코드에서는 집단 연산 시간을 절감한 결과임을 확인할 수 있었다.

V. 결 론

본 연구에서는 RAMSES-HR5 시뮬레이션 코드의 핫스팟을 분석하고 개선하기 위해, Intel VTune Profiler를 이용하여 프로파일링 하였으며, 이를 바탕으로 코드 최적화를 수행하였다. RAMSES-HR5 코드 전반에 걸쳐 자원 소모가 큰 주요 루틴들과 함수들을 식별하고, 해당 구간에서의 CPU 사용률, 메모리 접근 패턴, 및 MPI 통신대기 시간 등을 분석하였다. 프로파일링 결과 핵심 핫스팟 요소들을 해결하기 위해, OpenMP 기반의 쓰레드 병렬화 기법과 MPI 통신 최적화 및 I/O 감소 전략을 코드에 적용하였다. 본 연구에서는 RAMSES-HR5 코드는 그대로 유지하면서도 불필요한 자원 소모를 줄이고 병렬 효율성을 높일 수 있다. 최적화 결과 RAMSES-HR5의 실행 성능은 기존 대비 약 10~15% 향상되었다. 특히 OpenMP 동적 작업 스케줄링을 도입한 부분에서는 쓰레드 간 작업 편중이 완화되어 각 코어의 유휴 시간을 감소시켰으며, MPI 통신 측면에서는 비동기식 통신 활용과 불필요한 동기화 감소를 통해 다중 노드 간 데이터 교환에 소요되는 시간이 단축되었음을 확인하였다. 또한 빈번하게 수행되던 파일 입출력 연산을 최소화함으로써 I/O로 인한 지연이 줄어들었다. 이와 같은 개선으로 전체 시뮬레이션 시간 단축뿐만 아니라 HPC 자원의 활용 효율도 함께 높아져, 동일한 계산 자원으로 더 고해상도 또는 대규모의 시뮬레이션을 수행할 수 있는 가능성이 확대되었다. 향후에는 메모리 접근 패턴 개선이나 GPU 가속 등 추가적인 최적화 기법을 적용하고, 보다 다양한 시나리오에서의 성능 검증을 수행함으로써 RAMSES-HR5 시뮬레이션 코드를 비롯한 과학 계산 코드의 성능을 향상하는 연구를 수행할 예정이다 [13-15].

REFERENCES

[1] MATTSON, Timothy G.; SANDERS, Beverly; MASSINGILL, Berna. Patterns for parallel programming. Pearson Education, 2004.

[2] CHAPMAN, Barbara; JOST, Gabriele; VAN DER PAS, Ruud. Using OpenMP: portable shared memory parallel programming. MIT press, 2007.

[3] Intel Corp. VTune Profiler Performance Analysis Cookbook, 2024.

[4] TEYSSIER, Romain. Cosmological hydrodynamics with adaptive mesh refinement-a new high resolution code called ramses. Astronomy & Astrophysics, 2002, 385.1: 337-364.

[5] DUBOIS, Yohan, et al. Dancing in the dark: galactic properties trace spin swings along the cosmic web. Monthly Notices of the Royal Astronomical Society, 2014, 444.2: 1453-1468.

[6] MARTIN, G., et al. Cosmic reflections I: the structural diversity of simulated and observed low-mass galaxy analogues. Monthly Notices of the Royal Astronomical Society, 2025. staf1092.

[7] PRESS, William H. Numerical recipes 3rd edition: The art of scientific computing. Cambridge university press, 2007.

[8] BERGER, Marsha J.; COLELLA, Phillip. Local adaptive mesh refinement for shock hydrodynamics. Journal of computational Physics, 1989, 82.1: 64-84.

[9] DEMMEL, James W. Applied numerical linear algebra. Society for Industrial and Applied Mathematics, 1997.

[10] HOCKNEY, Roger W.; EASTWOOD, James W. Computer simulation using particles. crc Press, 2021.

[11] GROPP, William; SNIR, Marc. MPI: the complete reference. The MPI-2 extensions. MIT press, 1998.

[12] BALAY, Satish, et al. Petsc users manual (rev. 3.13). Argonne National Lab.(ANL), Argonne, IL (United States), 2020.

[13] KULIKOV, Igor M., et al. AstroPhi: A code for complex simulation of the dynamics of astrophysical objects using hybrid supercomputers. Computer Physics Communications, 2015, 186: 71-80.

[14] SPRINGEL, Volker. The cosmological simulation code GADGET-2. Monthly notices of the royal astronomical society, 2005, 364.4: 1105-1134.

[15] YANG, Wangdong, et al. Performance optimization using partitioned SpMV on GPUs and multicore CPUs. IEEE Transactions on Computers, 2014, 64.9: 2623-2636.

저 자 소 개

정현미(정회원)

2010년 한남대학교 컴퓨터공학과 석사 졸업
2014년 한남대학교 컴퓨터공학과 학과 박사 졸업
2012년~현재 한국과학기술연구원 슈퍼컴퓨팅기술개발센터 선임연구원
<주관심분야 : HPC, 이기종컴퓨팅 , 병렬컴퓨팅, 클라우드 >

이현조(정회원)

2008년 전북대학교 컴퓨터공학과 석사 졸업
2014년 전북대학교 컴퓨터 공학과 박사 졸업
2015년~2016년 한국과학기술정보연구원 선임연구원
2017년~현재 : 한국농수산대학교 연구원
<주관심분야 : 병렬 질의처리, 암호화 빅데이터, AI >

정기문(정회원)

2001년 전남대학교 전산통계학 석사 졸업
2009년 전남대학교 정보보호학 박사 졸업
2001~2004 한국정보보호진흥원 연구원
2004~2005년 국가정보원 연구원
2005년~현재 한국과학기술정보연구원 슈퍼컴퓨팅기술개발센터 책임연구원
<주관심분야 : HPC 클라우드, 이기종 컴퓨팅, 클라우드 보안 >

채철주(종신회원)

2009년 한남대학교 컴퓨터공학과 박사 졸업
2009년~2013년 한국전자통신연구원 선임연구원
2013년~2016년 한국과학기술정보연구원 선임연구원
2016년~현재 한국농수산대학교 교양학부 교수
<주관심분야 : 빅데이터, 인공지능, 디지털농업>