ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Analisi e sviluppo di middleware DDS per la gestione dei consumi in sistemi HPC

Relatore
Prof. Andrea Bartolini

Candidato
Giacomo Madella

Ottobre 2023

Abstract

Indice

Introduzione

Nei settori come ricerca scientifica, simulazione, l'analisi dei dati, la progettazione e ingegneria, dove la capacità di eseguire analisi dettagliate e computazioni intensive è essenziale per il progresso scientifico e tecnologico, l'elaborazione ad alte prestazioni (HPC, High-Performance Computing) rappresenta una risorsa vitale. Questa forma calcolo si distingue per la sua capacità di sfruttare architetture hardware specializzate e tecnologie software avanzate per affrontare compiti computazionali di elevata complessità in tempi ridotti, spingendo i limiti delle prestazioni e dell'efficienza computazionale.Gli HPC sono progettati per affrontare sfide che richiedono quantità massicce di calcoli, come simulazioni complesse, analisi dei dati su larga scala e ottimizzazione di progetti. Ovviamente tutto questo ha un costo, e in questo ultimo decennio si sono manifestate delle sempre crescenti richieste di capacità e potenza computazionale mentre le tecnologie ad esse associate si sono avvicinate ai propri limiti fisici. La gestione energetica e della potenza di sistemi HPC è diventata una delle principali preoccupazioni, non solo a causa dei costi monetari, ma anche per la progettazione di nuove generazioni di supercomputer e per la sostenibilità ambientale. Questo è dovuto in parte anche alla fine delle leggi di Denard e Moore, che avevano previsto una crescita continua della capacità di calcolo e della densità dei transistor nei circuiti integrati Tali leggi, che avevano guidato l'industria informatica per decenni, hanno perso progressivamente la loro validità poiché la fine dell'innovazione dei transistor ha comportato un consumo energetico crescente al crescere della velocità e capacità computazionale, rendendo sempre più difficilmente il mantenimento di questi sistemi. Con Power Management si definisce una stack software che gestisce la potenza e l'energia dei sistemi HPC e standardizza le interfacce tra diversi livelli di componenti software. Uno degli aspetti chiave di un PowerStack è definire una visione globale per la gestione energetica, estensibile ed in grado di ottimizzare la metrica di efficienza desiderata, consapevole dell'applicazione, in modo da poter scambiare potenza, energia e tempo di risoluzione al fine di ottimizzare l'efficienza di un'applicazione HPC. Il secondo aspetto è definire un'interfaccia standard per interagire con i controlli software e hardware di ottimizzazione su sistemi HPC di diversi fornitori. Mentre sono state proposte diverse tecniche per la gestione della potenza e dell'energia, la maggior parte di queste tecniche è stata ideata per

soddisfare singole esigenze di uno specifico centro di calcolo ad alte prestazioni o per obiettivi di ottimizzazione mirati su un insieme di problemi. Un recente studio [22] condotto dal gruppo di lavoro EEHPC [9] ha concluso che la maggior parte di tali tecniche manca di una consapevolezza globale necessaria per ottenere le migliori prestazioni di sistema e throughput. Inoltre, ciascuna tecnica tende a migliorare la gestione di potenza ed energia per un sottoinsieme diverso dell'hardware del sito o del sistema e a diverse granularità (spesso in conflitto). Sfortunatamente, le tecniche esistenti non sono state progettate per coesistere simultaneamente su un unico sito e cooperare nella gestione in modo efficiente. L'obiettivo finale sarebbe infatti quello di collegare gli strumenti disponibili utilizzando un approccio distribuito, sfruttando il potenziale del Data Distribution Service (DDS)[4] e del Real-Time Publish-Subscribe (RTPS).

1.1 Contributi

- Definire le interfacce tra questi livelli per tradurre gli obiettivi a ciascun livello in azioni al livello inferiore adiacente. - Promuovere l'ottimizzazione end-to-end attraverso diversi livelli del PowerStack.

Power management

2.1 Stato dell'arte

Come mostrato nella Figura 1, il Power Management è collegato: (i) on-chip ai Power Management (gestendo il consumo energetico e le prestazioni degli elementi di elaborazione principali) e ai sensori (monitorando il processo, la temperatura e la tensione degli elementi di elaborazione principali); (ii) off-chip ai Moduli Regolatori di Tensione (VRM) che alimentano il chip, gli altri componenti a bordo e il Controller di Gestione della Scheda (BMC).

La gestione della potenza utilizza questi componenti hardware e connessioni per supportare un insieme di servizi in-band e out-of-band.

I servizi in-band vengono forniti alle applicazioni e ai sistemi operativi in esecuzione negli elementi di elaborazione del chip e sono composti da: (i) governor dedicati alla potenza e telemetria correlata alla potenza a livello di sistema operativo; (ii) un'interfaccia dedicata per consentire alle applicazioni e ai tempi di esecuzione del modello di programmazione di specificare suggerimenti e prescrizioni per la gestione della potenza; (iii) un'interfaccia dedicata al Sistema e alla Gestione delle Risorse per supportare il capping della potenza a livello di CPU e nodo, nonché per gestire il compromesso tra Throughput ed Efficienza Energetica. I servizi out-of-band vengono forniti all'amministratore di sistema e agli strumenti di gestione del sistema tramite il Controller di Gestione della Scheda (BMC). Questi servizi consistono nella telemetria di potenza out-of-band, nel capping di potenza a livello di sistema e nella affidabilità e assistenza.

2.1.1 Servizi In-Band

Il Power Management condivide una regione di memoria interna con lo spazio degli indirizzi I/O degli elementi di elaborazione. Questa interfaccia consente all'OS di accedere periodicamente a un insieme di strutture dati di stato contenenti lo stato del Power Management, le statistiche e il consumo energetico dei diversi binari di tensione e componenti. Queste informazioni possono essere utilizzate e accessibili dalle applicazioni e

dagli utenti per monitorare in modo dettagliato l'energia consumata dalle applicazioni, consentendo la consapevolezza energetica.

2.1.2 Servizi Out-of-Band

Oltre alla politica di gestione della potenza e ai servizi In-Band, il Power Management si interfaccia con il BMC per supportare servizi Out-of-Band. Questi includono la telemetria dettagliata sullo stato di potenza e prestazioni del chip, il Power Management a livello di chip e a livello di sistema e la segnalazione di errori e guasti nel chip e nei processi principali.

2.1.3 Interfacce di alto livello

2.2 Componenti PowerStack

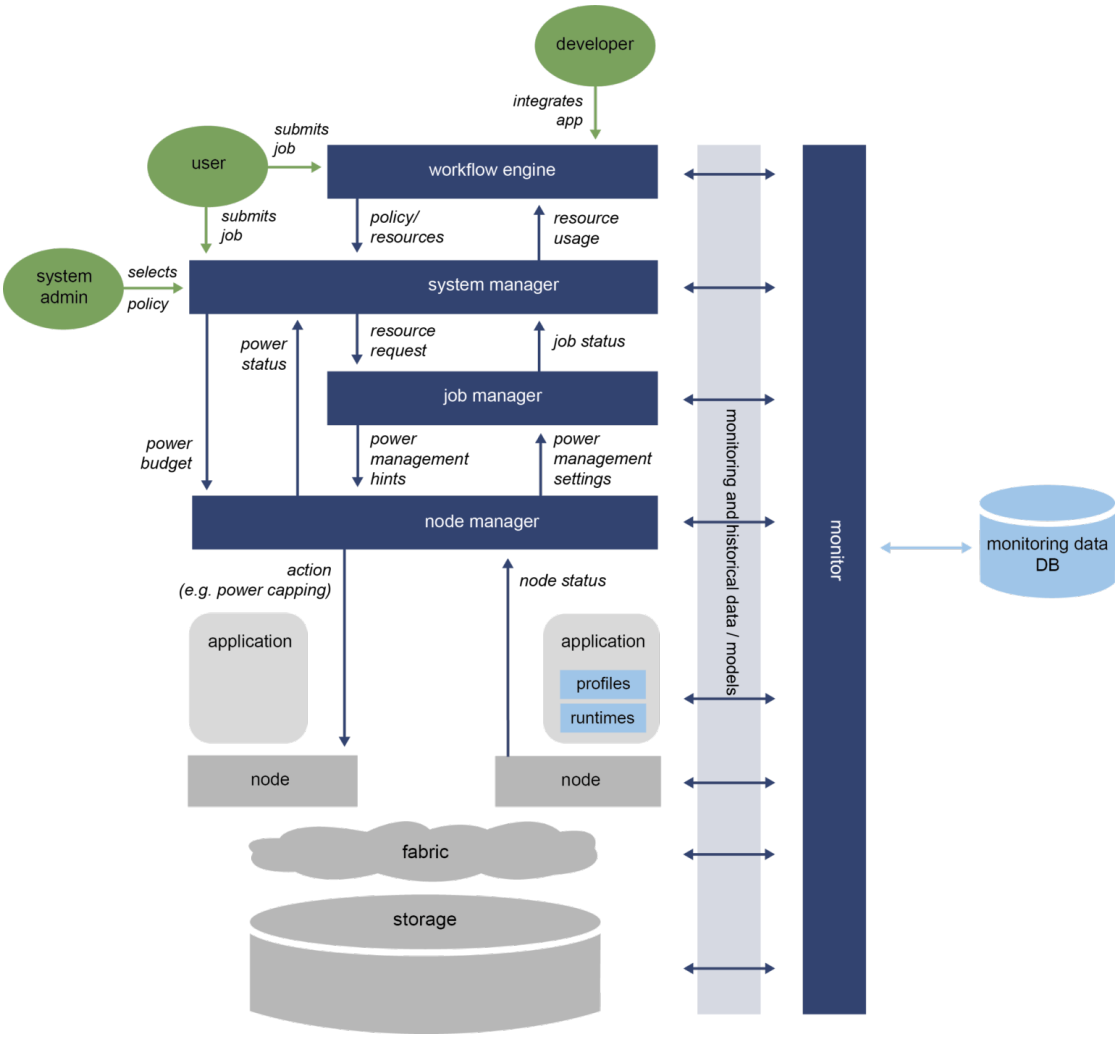
2.2.1 Workflow engine

The workflow engine analyses the dependencies and resource requirements of each workflow and decides on how to break the workflow into specific jobs that will be fed to the system manager.

Job schedulers allow high-performance computing users to efficiently share the computing resources that comprise an HPC system. Users submit batch jobs into one or more batch queues that are defined within the job scheduler. The job scheduler examines the overall set of pending work waiting to run on the computer and makes decisions about which jobs to place next onto the computational nodes within the computer. Generally speaking, the job scheduler attempts to optimize some characteristic such as overall system utilization or fast access to resources for some subset of batch jobs within the computing center's overall workload. The various queues that are defined within the job scheduler may be designated as having higher or lower priorities and may be restricted to some subset of the center's users, thus allowing the job scheduler to understand distinctions of importance of certain jobs within the overall workflow.

2.2.2 System Manager

Receives as input a set of jobs to be scheduled within the system and indicatively decides upon when to schedule each job, to which specific compute nodes to map it, and under which power budget or setting. For this, it constantly monitors and records power and energy telemetry data, and controls power budgets/settings and/or user fairness


developer
integrates app
user
submits job
workflow engine
submits job
policy/ resources
resource usage
system admin
selects policy
system manager
power status
resource request
job status
job manager
power budget
power management hints
power management settings
node manager
action (e.g. power capping)
node status
application
application
profiles
runtimes
node
node
fabric
storage
monitoring and historical data / models
monitor
monitoring data DB

2.2.3 Job Manager

By analysing the profiles, Job Manager decides the target power management knob (CPU power cap, CPU clock frequency scaling or any others) as well as performs code tuning as an option.

2.2.4 Node Manager

The node manager provides access to node-level hardware controls and monitors. Moreover, the node manager implements processor level and node level power management policies, as well as preserving the power integrity, security and safety of the node.

2.2.5 Monitor

The monitor is responsible for collecting in-band and out-of-band data for performance, resource utilization, status, power, energy, with minimal footprint, collecting, aggregating, and analysing various metrics, and pushing necessary real-time data to the other entities.

REGALE

3.1 Power Stack

3.2 Problematica

DDS & RTPS

DDS (Data Distribution Service)[1] e RTPS (Real-Time Publish-Subscribe)[2] costituiscono due soluzioni fondamentali nel campo delle comunicazioni distribuite e real-time. Queste tecnologie svolgono un ruolo importante nella la trasmissione di dati tra dispositivi e applicazioni interconnesse, rivestendo particolare importanza in scenari complessi come i sistemi embedded, in IoT e applicazioni ad alte prestazioni come l'HPC (High-Performance Computing).

4.1 Implementazione usasta

DDS e RTPS sono dei protocolli di comunicazione per specifici casi di utilizzo. Ci sono state diverse implementazioni di questi protocolli da diversi società e organizzazioni, come:

- FastDDS (eprosima)
- CycloneDDS (Oracle)
- ConnextDDS
- GurumDDS

E tante altre. In tutti i successivi capitoli verrà preso come riferimento FastDDS, una implementazione specifica per le comunicazione Real-Time.

4.2 DDS

Data Distribution Service è un protocollo di comunicazione incentrato sullo scambio di dati per sistemi distribuiti. Questo si basa su modello chiamato Data-Centric Publish Subscribe (DCPS) I principali attori che vengono coinvolti sono:

- Publisher: responsabile della creazione e configurazione dei DataWriter. Il DataWriter è l'entità responsabile della pubblicazione effettiva dei messaggi. Ciascuno avrà un Topic assegnato sotto il quale vengono pubblicati i messaggi;
- Subscriber: responsabile di ricevere i dati pubblicati sotto i topic ai quali si iscrive. Serve uno o più oggetti DataReader, che sono responsabili di comunicare la disponibilità di nuovi dati all'applicazione;
- Topic: collega i DataWriter con i DataReader. È univoco all'interno di un dominio DDS;
- Dominio: utilizzato per collegare tutti i publisher e subscriber appartenenti a una o più domini di appartenenza, che scambiano dati sotto diversi topic. Il DomainParticipant funge da contenitore per altre entità DCPS, e svolge anche la funzione di costruttore di entità Publisher, Subscriber e Topic fornendo anche servizi di QoS;
- Partizione: costituisce un isolamento logico di entità all'interno dell'isolamento fisico offerto dal dominio;

Inoltre DDS definisce le cosidette Qualità di Servizio (QoS policy) che servono configurare il comportamento di ognuno di questi attori.

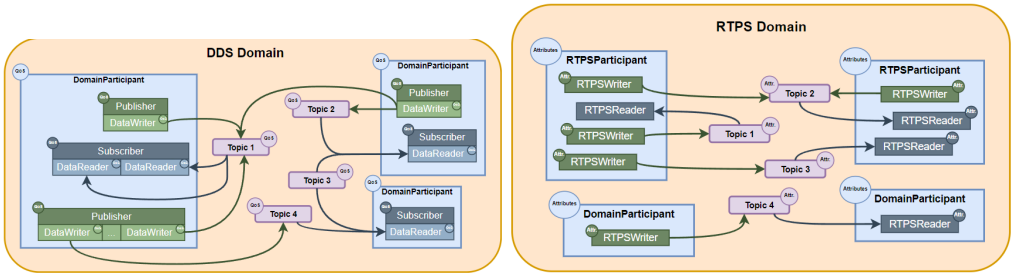
4.3 RTPS

Real-Time Publisher Subscribe protocol è un protocollo-middleware utilizzato da DDS per gestire la comunicazione su diversi protocolli di rete come UDP/TCP e Shared Memory. Il suo principale scopo è quello di inviare messaggi real-time, con un approccio best-effort e cercando di massimizzare l'efficienza. E' inoltre progettato per fornire strumenti per la comunicazione unicast e multicast. Le principali entità descritte da RTPS sono:

- RTPSWriter: endpoint capace di inviare dati;
- RTPSReader: endpoint abilitato alla ricezione dei dati;

Ereditato da DDS anche RTPS ha la concezione di Dominio di comunicazione e come questo, le communcazioni a livello di RTPS girano attorno al concetto di Topic prima definito. L'unità di comunicazione è chiamata **Change** che rappresenta appunto un cambiamento sui dati scritti sotto un certo topic. Ognuno degli attori registra questi *Change* in una struttura dati che funge da cache. In particolare la sequenza di scambio è:

1. il *change* viene aggiunto nella cache del RTPSWriter;
2. RTPSWriter manda questa *change* a tutti gli RTPSReader che conosce;
3. quando RTPSReader riceve il messaggio, aggiorna la sua cache con il nuovo *change*.



(a) Schema DDS **(b)** Schema RTPS

Figura 4.1. Confronto tra architettura DDS e RTPS

Casi di studio e valutazioni

In questa sezione verranno riportati i casi studio e i test effettuati sul framework. Tutti questi casi studio sono stati eseguiti su un sistema HPC Galielo-100 Cineca con le specifiche riportate nella tabella sotto

Parameter	**Value**
Number of nodes used	3
Processor	Intel CascadeLake 8260
Number of sockets per node	2
Number of cores per socket	24
Memory size per node	384 GB
Interconnect	Mellanox Infiniband 100GbE
OS	CentOS Linux
MPI	Open MPI 4.1.1

5.1 Test

Sono stati svolti diversi test al fine di creare un modello per l'utilizzo e la caratterizzazione di DDS, ed in particolare è stata usato FastDDS, all'interno di sitemi HPC, nel contesto del Power Management.

In particolare nel Publisher 4.2 prima e dopo la chiamata a funzione di write() si sono presi i valori time-invio, istruzioni, e TSC, mentre al lato ricevente, di Subscriber 4.2 è stato preso il tempo al momento dell'arrivo del messaggio. In questo modo si sono misurati i seguenti parametri:

- Tempo solo invio
- Tempo invio-ricezione
- Perf-Event Instructions
- TSC (read_tsc)

5.1.1 Sincronizzazione orologi su nodi diversi

Per ottenere risultati attendibili sulla metrica del tempo è stato necessario sincronizzare i nodi utilizzati prima di poter far partire i test. Per farlo è stato usata una funzione *CLOCK_MONOTONIC* che rappresenta un *un orologio non impostabile a livello di sistema che rappresenta il tempo monotono da un punto non specificato nel passato. Su Linux, quel punto corrisponde al numero di secondi di esecuzione del sistema da quando è stato avviato. L'orologio CLOCK_MONOTONIC non è influenzato da salti discontinui nell'ora del sistema, ma è influenzato dalle regolazioni incrementali eseguite da NTP.* Il problema che si è presentato, è che avendo nodi diversi su cui far eseguire i test, per provare ad esempio nel modo più affidabile i protocolli di trasporto, è stato necessario implementare delle MPI_Barrier prima di diverse esecuzioni di *clock_gettime(CLOCK_MONOTONIC)*. Di seguito sono stati riportati i grafici del risultato ottenuto.

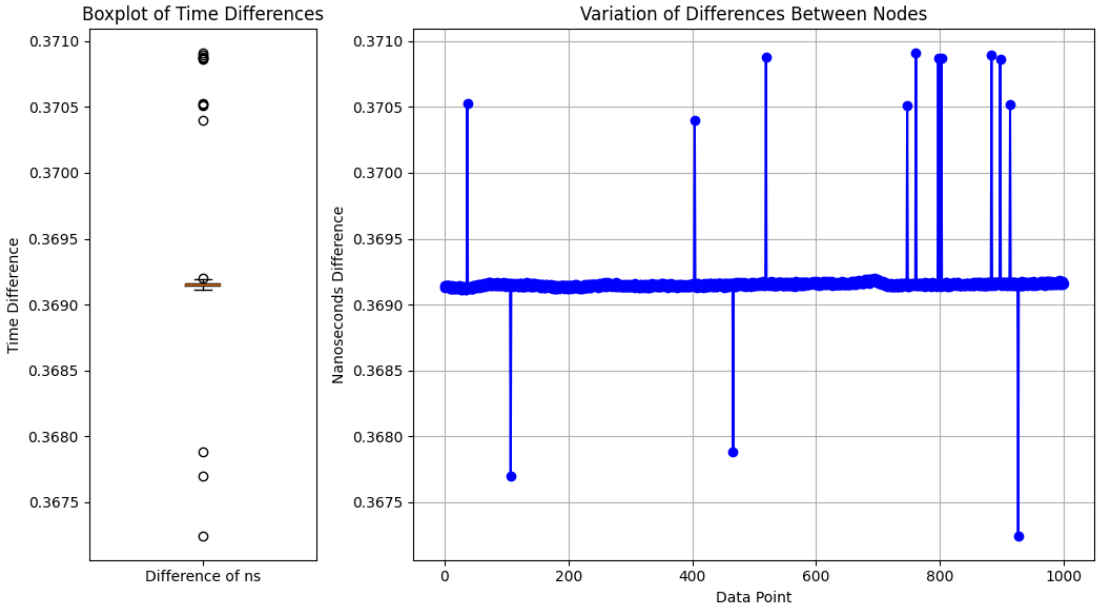


Figura 5.1. Scostamento del tempo su nodi divers

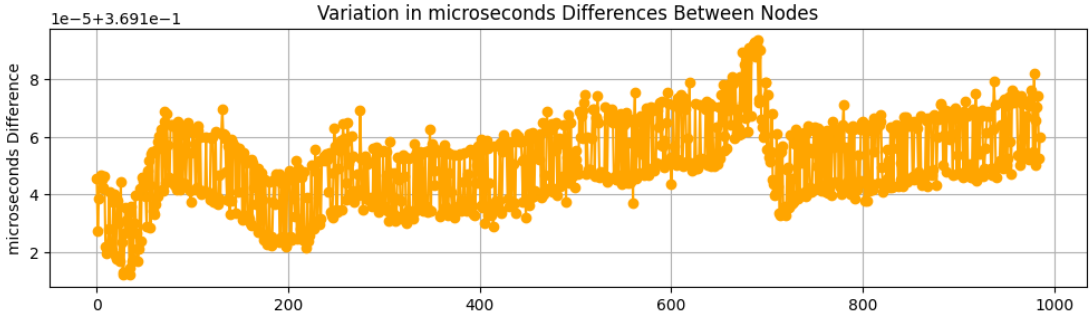


Figura 5.2. Scostamento senza outliers

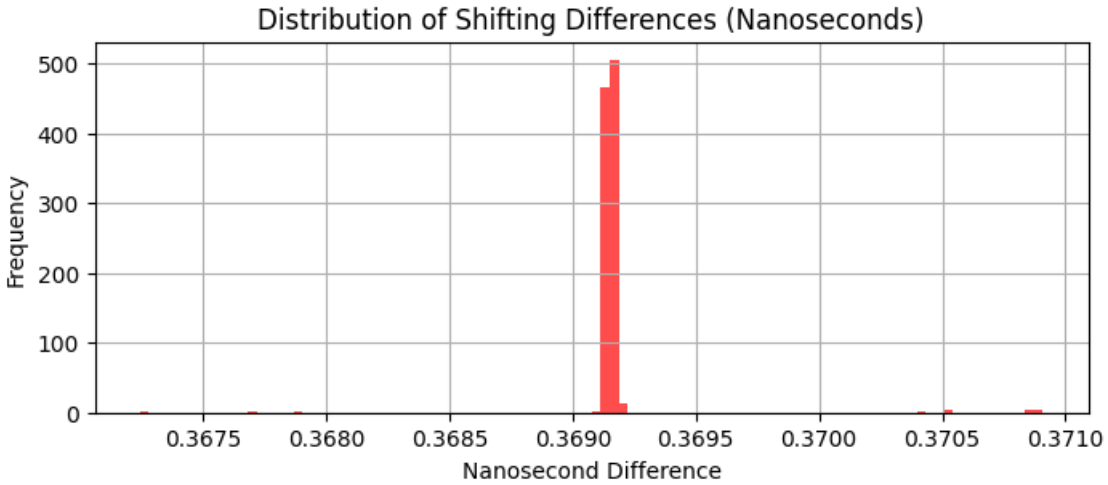


Figura 5.3. Distribuzione delle differenze

Come possibile vedere nella figura 5.1 nonostante le mpi_barrier, sono presenti degli scostamenti di tempo tra 2 nodi durante diversi test effettuati (in particolare 1000), e si è scelto di utilizzare il valore modale di questa differenza,sulla base del quale, si sono elaborati tutti i dati successivi.

5.1.2 Impatto del numero di sub in un dominio

5.1.3 Test1

Nel primo test si è valutata la differenza di diversi protocolli di trasporto

5.1.4 Test2

5.1.5 Test3

5.2 Risultati

5.3 Modello

5.4 Scheletro componenti

Nel seguente capitolo viene stilato uno scheletro dei componenti con i relativi topic usati al fine di dare una visione completa e aggiuntiva rispetto al modello precedentemente stilato

DUMMIES	
NAME	**USED**
NODE MANAGER DUMMY	P 0 default Monitor_report_job_telemetry P 0 default Monitor_report_node_telemetry P 0 default Monitor_report_cluster_telemetry
JOB SCHEDULER DUMMY	P 0 default SystemPowerManager_get P 0 default SystemPowerManager_set S 0 default SystemPowerManager_get_reply S 0 default SystemPowerManager_set_repl
JOB MANAGER DUMMY	S 0 default NodeManager_get P 0 default NodeManager_get_reply

SERVERS	
NAME	**OFFERED**
NODE MANAGER	S 0 default NodeManager_get S 0 default NodeManager_set P 0 default NodeManager_get_reply P 0 default NodeManager_set_reply
SYSTEM POWER MANAGER	S 0 default SystemPowerManager_get S 0 default SystemPowerManager_set P 0 default SystemPowerManager_get_reply P 0 default SystemPowerManager_set_reply
MONITOR	S 0 default Monitor_report_job_telemetry S 0 default Monitor_report_node_telemetry S 0 default Monitor_report_cluster_telemetry

Conclusioni

È stato dimostrato come un framework di comunicazione DDS può essere usato all'interno di un Power-Stack per la gestione di energia in sistemi HPC vincolati dalla potenza al fine di affrontare il problema della limitazione energetica.

Bibliografia

[1] Object Management Group. *Data Distribution Service*. 2004. URL: https://www.omg.org/spec/DDS/1.0.

[2] Object Management Group. *DDS Interoperability Wire Protocol*. 2008. URL: https://www.omg.org/spec/DDSI-RTPS/2.0.