01

Task C2

How task C2 is processed?

02

Remove Fetch

How can we avoid fetch operations using shared storage?

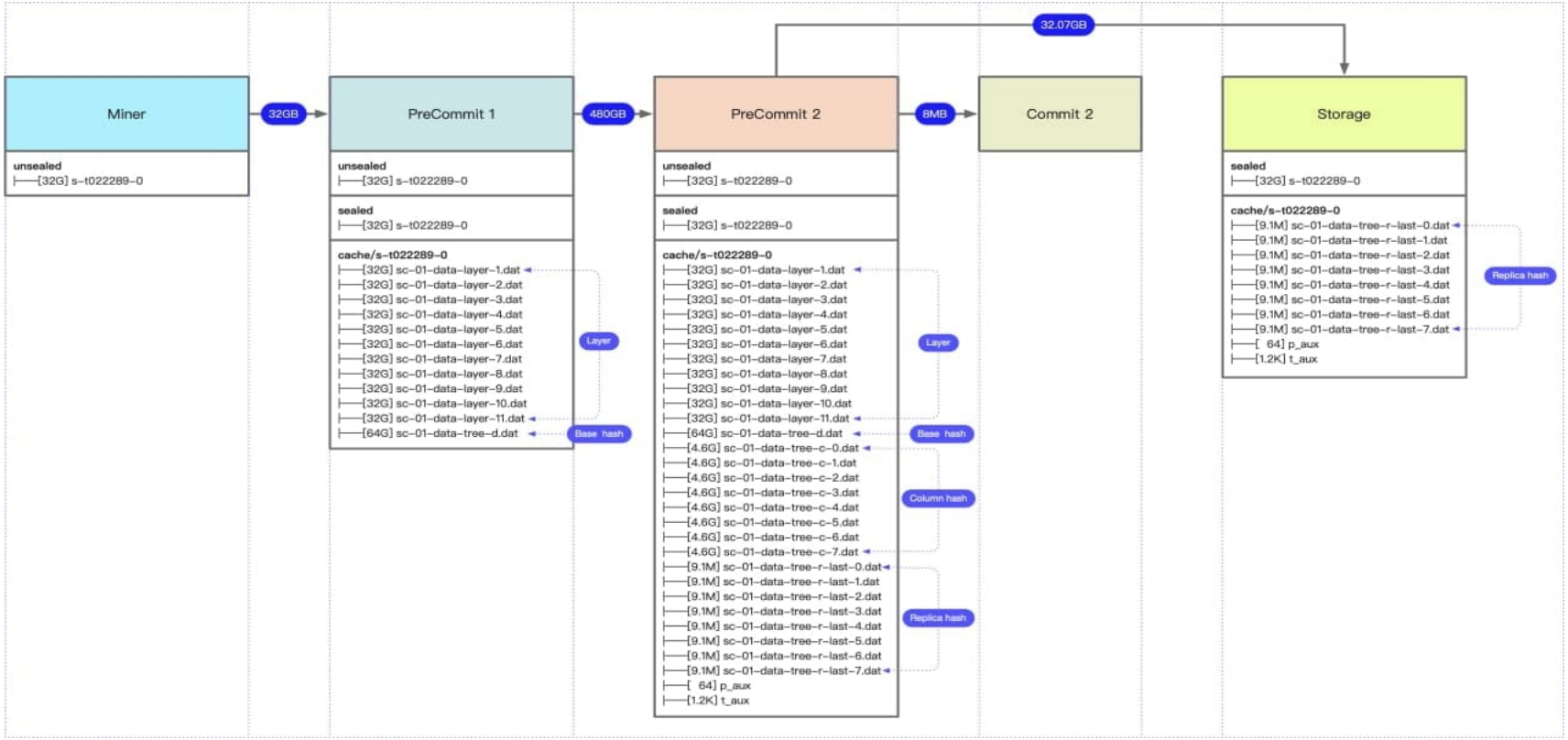
03

RDMA

RDMA solution

Sealing Job tasks

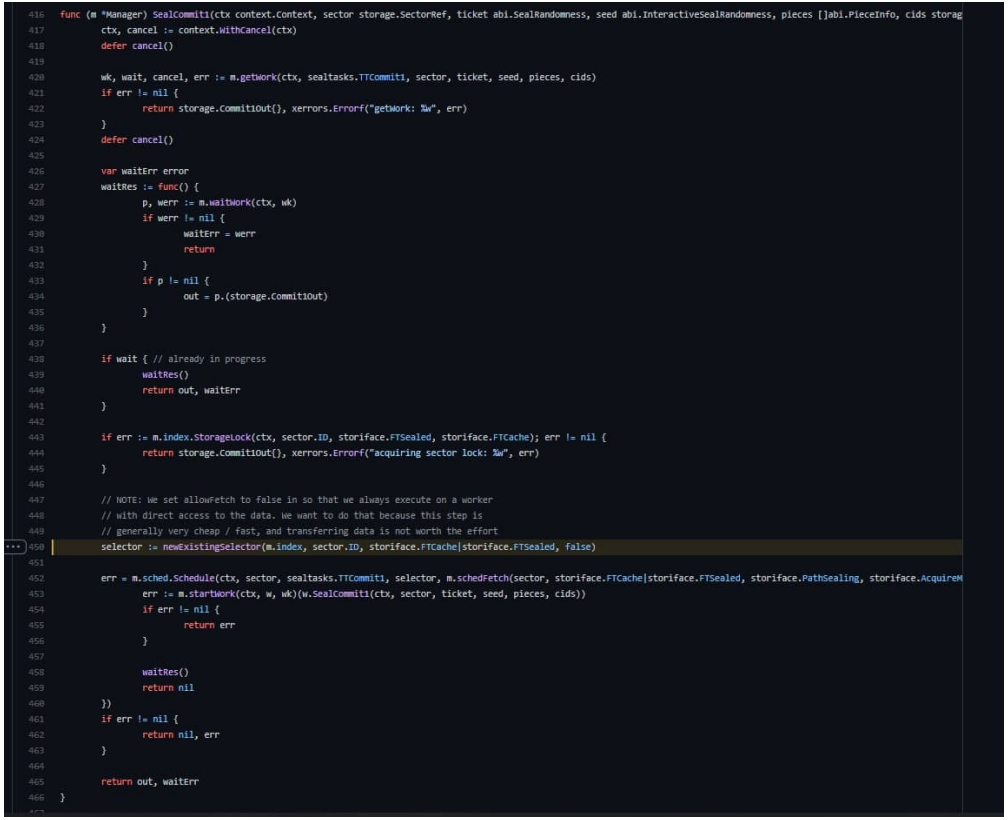
Sealing Jobs storage usage



How task C2 is processed?

- During the previous meeting, there were doubts about the way C2 Task was handled. Our guess is that, since the C2 Worker does not require storage, it can directly access the PC2 Worker's storage and completed tasks.
- On the same worker that completed PC2, there is a small task called C1 that continues the process. After the C1 job has finished, a 20MB file is created and sent to the C2 worker using **JSONRPC**.
- The data has to be fetched, but compared to PC1 and PC2, the data transfer is very small.

C1 → C2 Data Transfer



```
416 func (m *Manager) SealCommit1(ctx context.Context, sector storage.SectorRef, ticket abi.SealRandomness, seed abi.InteractiveSealRandomness, pieces []abi.PieceInfo, cids storag
417         ctx, cancel := context.WithCancel(ctx)
418         defer cancel()
419
420         wk, wait, cancel, err := m.getWork(ctx, sealtasks.TTCommit1, sector, ticket, seed, pieces, cids)
421         if err != nil {
422                 return storage.Commit1Out{}, xerrors.Errorf("getWork: %w", err)
423         }
424         defer cancel()
425
426         var waitErr error
427         waitRes := func() {
428                 p, werr := m.waitWork(ctx, wk)
429                 if werr != nil {
430                         waitErr = werr
431                         return
432                 }
433                 if p != nil {
434                         out = p.(storage.Commit1Out)
435                 }
436         }
437
438         if wait { // already in progress
439                 waitRes()
440                 return out, waitErr
441         }
442
443         if err := m.index.StorageLock(ctx, sector.ID, storiface.FTSealed, storiface.FTCache); err != nil {
444                 return storage.Commit1Out{}, xerrors.Errorf("acquiring sector lock: %w", err)
445         }
446
447         // NOTE: We set allowFetch to false in so that we always execute on a worker
448         // with direct access to the data. We want to do that because this step is
449         // generally very cheap / fast, and transferring data is not worth the effort
450         selector := newExistingSelector(m.index, sector.ID, storiface.FTCache|storiface.FTSealed, false)
453
452         err = m.sched.Schedule(ctx, sector, sealtasks.TTCommit1, selector, m.schedFetch(sector, storiface.FTCache|storiface.FTSealed, storiface.PathSealing, storiface.AcquireM
453                 err := m.startWork(ctx, w, wk)(w.SealCommit1(ctx, sector, ticket, seed, pieces, cids))
454                 if err != nil {
455                         return err
456                 }
457
458                 waitRes()
459                 return nil
460         })
461         if err != nil {
462                 return nil, err
463         }
464
465         return out, waitErr
466 }
```

Simulating DevNet environment with Docker

Docker-compose for run lotus-daemon , lotus-miner

```
version: "3.1"

services:
    lotus-dev:
        container_name: lotus
        image: lotus-dev
        user: root
        network_mode: host
        privileged: true
        environment:
            - LOTUS_PATH=/root/devnet/.lotusDevnet
            - LOTUS_MINER_PATH=/root/devnet/.lotusminerDevnet
            - LOTUS_SKIP_GENESIS_CHECK=_yes_
            - LOTUS_MINER_JOB_LOG_PATH=/root
        command: >
            bash -c "lotus fetch-params 2048 &&
            lotus-seed pre-seal --sector-size 2KiB --num-sectors 2 &&
            lotus-seed genesis new /root/localnet.json &&
            lotus-seed genesis add-miner /root/localnet.json /root/.genesis-sectors/pre-seal-t01000.json &&
            nohup lotus daemon --lotus-make-genesis=/root/devgen.car --genesis-template=/root/localnet.json --bootstrap=false >/root/
            lotus-daemon.log 2>&1 & sleep 20 &&
            lotus wallet import --as-default /root/.genesis-sectors/pre-seal-t01000.key &&
            lotus-miner init --no-local-storage --genesis-miner --actor=t01000 --sector-size=2KiB --pre-sealed-sectors=/root/.genesis-sectors
            --pre-sealed-metadata=/root/.genesis-sectors/pre-seal-t01000.json --nosync &&
            nohup lotus-miner run --nosync > /root/lotus-miner.log 2>&1
            "
        volumes:
            - storage1:/root/devnet/storage1
            - storage2:/root/devenet/storage2
        tty: true
        stdin_open: true

volumes:
    storage1:
    storage2:
```

Simulating DevNet environment with Docker

Simulating DevNet Steps

1

Docker Image file

In the Docker image file, we have created an automated script that retrieves the code from the specified repository. The script fetches the source from the repository, compiles and exports it, allowing a developer to test the latest changes.

2

Lotus Devnet Sector Config

As soon as the Docker image can be used, we start lotus daemon and miner with 2k sectors that allow us to use smaller sectors for testing.

3

Config Miner Layout

Simulating a large set-up requires running multiple workers in docker containers on separate physical and virtually based servers. they are all part of the same miner.

4

Create deals and task

Scheduler can be used to assign tasks to workers after the environment is ready. Our tool creates multiple deals and makes the miner accept them in order to provide tasks to workers.

Solution for Eliminating the "Fetch" method

To gain a better understanding of how Lotus handles shared storage, we modified some parts of the Lotus project, added some additional logs to the source code, and ran tests in two scenarios :

- Creating an environment with separated storage which is the default configuration for the FileCoin, the result ended with GET operation.
- This setup uses a Docker volume to simulate a shared storage environment with several workers and different task support, with each worker doing a specific task and sending the process on to the next worker to complete. As they have shared storage, there wasn't a GET operation.

Tools & Configuration

Tools :

- Docker
- Sector 2 Kb

Configuration :

- Miner (AP, PC1 tasks)
- Worker (PC2, C2 tasks)

```
[Sealing]
  MaxWaitDealsSectors = 6
  MaxSealingSectors = 600
  MaxSealingSectorsForDeals = 7
  WaitDealsDelay = "3h0m0s"
  AlwaysKeepUnsealedCopy = true
  FinalizeEarly = true
  BatchPreCommits = true
  MaxPreCommitBatch = 15
  MinPreCommitBatch = 5
  PreCommitBatchWait = "0h15m0s"
  PreCommitBatchSlack = "5h0m0s"
  AggregateCommits = true
  MinCommitBatch = 5
  MaxCommitBatch = 15
  CommitBatchWait = "0h20m0s"
  CommitBatchSlack = "5h0m0s"
  TerminateBatchMax = 100
  TerminateBatchMin = 1
  TerminateBatchWait = "5m0s"
```

Scenario 1: Separate Storage

Commands

Initialize miner :

```
lotus-miner init --genesis-miner --actor=t01000 --sector-size=2KiB --pre-sealed-sectors=/root/.genesis-sectors --pre-sealed-metadata=/root/.genesis-sectors/pre-seal-t01000.json -nosync
```

Initialize worker:

```
lotus-worker run --listen "0.0.0.0:3455"
```

Scenario 1: Separate Storage

Worker Result

```
2021-07-30T03:47:39.104-0400    INFO    main    lotus-seal-worker/main.go:336   Opening local storage; connecting to master
2021-07-30T03:47:39.106-0400    INFO    main    lotus-seal-worker/main.go:394   Setting up control endpoint at 127.0.0.1:3456
2021-07-30T03:47:39.107-0400    INFO    main    lotus-seal-worker/main.go:497   Making sure no local tasks are running
2021-07-30T03:47:39.142-0400    INFO    main    lotus-seal-worker/main.go:520   Worker registered successfully, waiting for tasks


Step1 : paths {{0 0}   } , stores {{0 0}   }



Step2 : apaths {{0 0} /root/.lotusworker/unsealed/s-t01000-2  } , ids {{0 0} 4c93ce3d-c66d-4d08-ac7c-0670a97405b9  }



Step3 : pathType sealing , PathStorage storage



Step4 : fileTypes unsealed

2021-07-30T03:50:02.087-0400    INFO    stores  stores/remote.go:242    Fetch http://0.0.0.0:2345/remote/unsealed/s-t01000-2 -> /root/.
lotusworker/unsealed/fetching/s-t01000-2

Step5 : dest /root/.lotusworker/unsealed/s-t01000-2 , storageID 4c93ce3d-c66d-4d08-ac7c-0670a97405b9, url http://0.0.0.0:2345/remote/unsealed/
s-t01000-2

2021-07-30T03:50:02.091-0400    INFO    stores  stores/remote.go:354    Delete http://0.0.0.0:2345/remote/unsealed/s-t01000-2


Step6 : paths {{0 0} /root/.lotusworker/unsealed/s-t01000-2  } , stores {{0 0} 4c93ce3d-c66d-4d08-ac7c-0670a97405b9  }

2021-07-30T03:50:02.092-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 2} 5} (e:1; a:0): {{0 0} /root/.
lotusworker/unsealed/s-t01000-2  }

Step7 : outname /root/.lotusworker/unsealed/fetching/s-t01000-2 , resp.header map[Accept-Ranges:[bytes] Content-Length:[2055] Content-Type:
[application/octet-stream] Date:[Fri, 30 Jul 2021 07:50:02 GMT] Last-Modified:[Fri, 30 Jul 2021 07:50:02 GMT]]

2021-07-30T03:50:02.088-0400    DEBUG   stores  stores/util_unix.go:30  move sector data    {"from": "/root/.lotusworker/unsealed/fetching/
s-t01000-2", "to": "/root/.lotusworker/unsealed/s-t01000-2"}
```

Scenario 1: Separate Storage

Tools & Configuration

Tools :

- Docker
- Sector 2 Kb
- Share storage with docker volume

Configuration :

- Miner
- Worker 77a69335 (C2, PC2 tasks)
- Worker 81346a03 (PC1 task)

```
[Sealing]
  MaxWaitDealsSectors = 6
  MaxSealingSectors = 600
  MaxSealingSectorsForDeals = 7
  WaitDealsDelay = "3h0m0s"
  AlwaysKeepUnsealedCopy = true
  FinalizeEarly = true
  BatchPreCommits = true
  MaxPreCommitBatch = 15
  MinPreCommitBatch = 5
  PreCommitBatchWait = "0h15m0s"
  PreCommitBatchSlack = "5h0m0s"
  AggregateCommits = true
  MinCommitBatch = 5
  MaxCommitBatch = 15
  CommitBatchWait = "0h20m0s"
  CommitBatchSlack = "5h0m0s"
  TerminateBatchMax = 100
  TerminateBatchMin = 1
  TerminateBatchWait = "5m0s"
```

```
[Storage]
  ParallelFetchLimit = 5
  AllowAddPiece = false
  AllowPreCommit1 = false
  AllowPreCommit2 = false
  AllowCommit = false
  AllowUnseal = true
```

Scenario 2: Shared Storage

Commands

Initialize miner without local storage :

```
lotus-miner init --no-local-storage --genesis-miner --actor=t01000 --sector-
size=2KiB --pre-sealed-sectors=/root/.genesis-sectors
--pre-sealed-metadata=/root/.genesis-sectors/pre-seal-t01000.json -nosync
```

Initialize workers without local storage :

```
lotus-worker run --no-local-storage --precommit1=false --precommit2=true --
commit=true --listen "0.0.0.0:3457"
```

Attach and Init mounted storage for miner :

```
lotus-miner storage attach --seal --init /root/devnet/storage1
lotus-miner storage attach --seal --init /root/devnet/storage2
```

Attach mounted storage for workers :

```
lotus-worker storage attach --seal /root/devnet/storage1
lotus-worker storage attach --seal /root/devnet/storage2
```

Scenario 2: Shared Storage

Workers Info

```
Session: 77a69335-5a26-4059-8b75-
2131318ba7e3
Enabled: true
Hostname: ip-172-31-36-231
CPUs: 8; GPUs: []
RAM: 15.46 GiB; Swap: 0 B
Reserved memory: 2.223 GiB
Task types: FIN GET UNS C1 C2 PC2 AP

bb56783f-faec-427a-9848-1bfb4059f0a4:
        Weight: 10; Use: Seal
        Local: /root/devnet/storage1
fac05ecc-8b6b-424f-a33c-e3eb6eb694a8:
        Weight: 10; Use: Seal
        Local: /root/devnet/storage2
```

```
Session: 81346a03-8370-4f98-a259-
c5781e0f0901
Enabled: true
Hostname: ip-172-31-36-231
CPUs: 8; GPUs: []
RAM: 15.46 GiB; Swap: 0 B
Reserved memory: 2.548 GiB
Task types: FIN GET UNS C1 PC1 AP

bb56783f-faec-427a-9848-1bfb4059f0a4:
        Weight: 10; Use: Seal
        Local: /root/devnet/storage1
fac05ecc-8b6b-424f-a33c-e3eb6eb694a8:
        Weight: 10; Use: Seal
        Local: /root/devnet/storage2
```

Scenario 2: Shared Storage

Worker 77a69335 Result

```
2021-07-30T03:04:36.280-0400    INFO    main    lotus-seal-worker/main.go:336   Opening local storage; connecting to master

2021-07-30T03:04:36.281-0400    INFO    main    lotus-seal-worker/main.go:394   Setting up control endpoint at 127.0.0.1:3457

2021-07-30T03:04:36.282-0400    INFO    main    lotus-seal-worker/main.go:497   Making sure no local tasks are running

2021-07-30T03:04:36.300-0400    INFO    main    lotus-seal-worker/main.go:520   Worker registered successfully, waiting for tasks

2021-07-30T03:27:04.103-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 2} 5} (e:6; a:0): {{0 0}  /root/
devnet/storage1/sealed/s-t01000-2 /root/devnet/storage1/cache/s-t01000-2}

2021-07-30T03:27:04.112-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 2} 5} (e:6; a:0): {{0 0}  /root/
devnet/storage1/sealed/s-t01000-2 /root/devnet/storage1/cache/s-t01000-2}

2021-07-30T03:27:04.139-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 4} 5} (e:6; a:0): {{0 0}  /root/
devnet/storage1/sealed/s-t01000-4 /root/devnet/storage1/cache/s-t01000-4}

2021-07-30T03:27:04.166-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 4} 5} (e:6; a:0): {{0 0}  /root/
devnet/storage1/sealed/s-t01000-4 /root/devnet/storage1/cache/s-t01000-4}

2021-07-30T03:27:39.798-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 3} 5} (e:6; a:0): {{0 0}  /root/
devnet/storage1/sealed/s-t01000-3 /root/devnet/storage1/cache/s-t01000-3}

2021-07-30T03:27:39.803-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 3} 5} (e:6; a:0): {{0 0}  /root/
devnet/storage1/sealed/s-t01000-3 /root/devnet/storage1/cache/s-t01000-3}
```

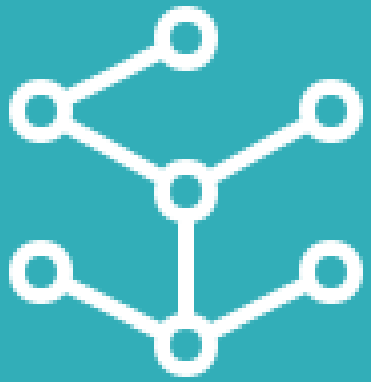
Worker 81346a03 Result

```
2021-07-30T03:22:09.065-0400    INFO    main    lotus-seal-worker/main.go:336   Opening local storage; connecting to master

2021-07-30T03:22:09.065-0400    INFO    main    lotus-seal-worker/main.go:394   Setting up control endpoint at 127.0.0.1:3458

2021-07-30T03:22:09.068-0400    INFO    main    lotus-seal-worker/main.go:497   Making sure no local tasks are running

2021-07-30T03:22:09.134-0400    INFO    main    lotus-seal-worker/main.go:520   Worker registered successfully, waiting for tasks

2021-07-30T03:27:04.058-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 2} 5} (e:1; a:0): {{0 0} /root/devnet/
storage1/unsealed/s-t01000-2  }

2021-07-30T03:27:04.066-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 2} 5} (e:1; a:6): {{0 0} /root/devnet/
storage1/unsealed/s-t01000-2 /root/devnet/storage1/sealed/s-t01000-2 /root/devnet/storage1/cache/s-t01000-2}

2021-07-30T03:27:04.111-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 4} 5} (e:1; a:0): {{0 0} /root/devnet/
storage1/unsealed/s-t01000-4  }

2021-07-30T03:27:04.118-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 4} 5} (e:1; a:6): {{0 0} /root/devnet/
storage1/unsealed/s-t01000-4 /root/devnet/storage1/sealed/s-t01000-4 /root/devnet/storage1/cache/s-t01000-4}

2021-07-30T03:27:39.772-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 3} 5} (e:1; a:0): {{0 0} /root/devnet/
storage1/unsealed/s-t01000-3  }

2021-07-30T03:27:39.779-0400    DEBUG   advmgr  sector-storage/worker_local.go:137  acquired sector {{1000 3} 5} (e:1; a:6): {{0 0} /root/devnet/
storage1/unsealed/s-t01000-3 /root/devnet/storage1/sealed/s-t01000-3 /root/devnet/storage1/cache/s-t01000-3}
```

Scenario 2: Shared Storage

Implementing scenario two using RDMA

In computing, remote direct memory access (RDMA) is a direct memory access from the memory of one computer into that of another without involving either one's operating system. This permits high-throughput, low-latency networking, which is especially useful in massively parallel computer clusters.



Bandwidth

High rate Bandwidth 10 ~ 400 GbE/s

Latency

1000 Nano Second latency

CPU

Reduces CPU utilization

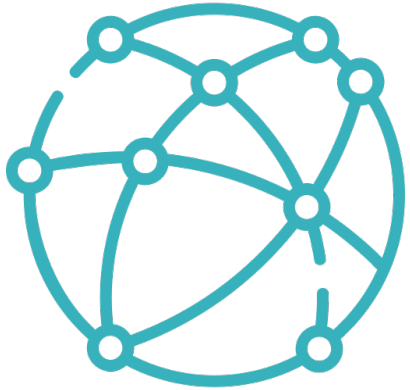
RDMA Advantages

Data transfer between servers and between server and storage without the involvement of the host CPU in the data path

Bandwidth 10 ~ 400 GbE/s

RDMA provides low latency through stack bypass and copy avoidance

Reduces CPU utilization



Reduces memory bandwidth bottlenecks and provides high bandwidth utilization

Optimized server utilization with high speed networking and CPU offload

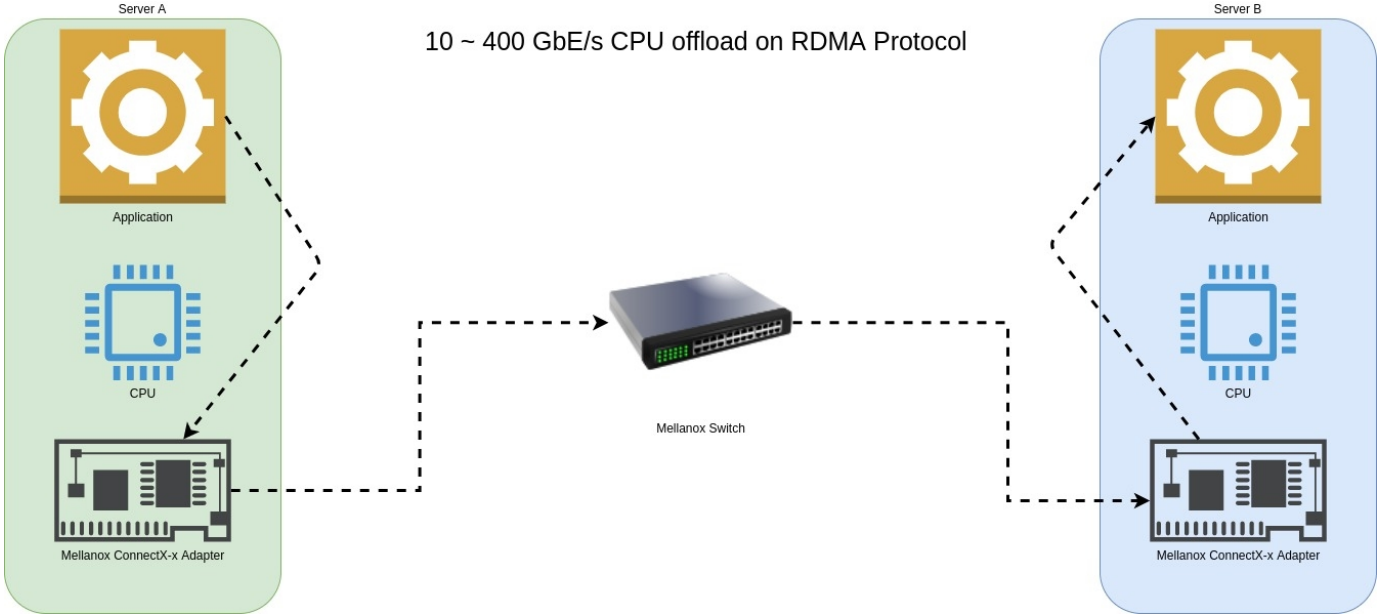
High availability and high reliability for compute, storage, and network

Increases total SSD utilization to over 90%

RMDA Protocol

CPU Offload

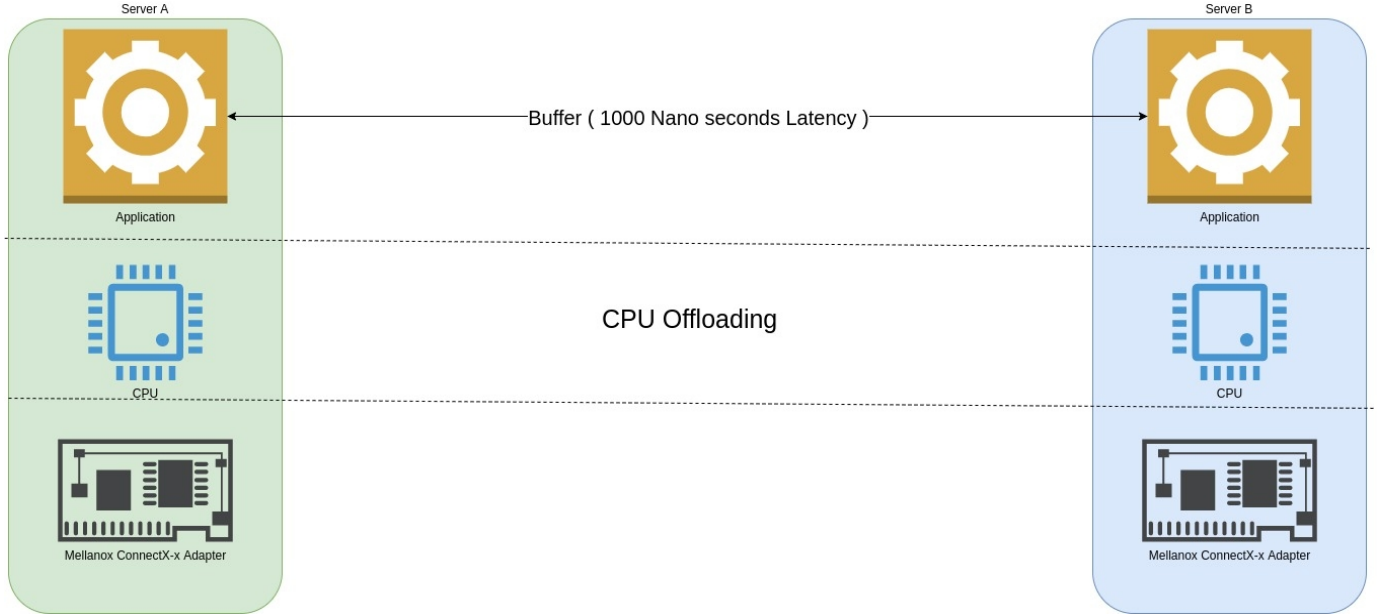
RDMA allows direct memory access from the memory of one node into that of another without involving either one's CPU.



RMDA Protocol

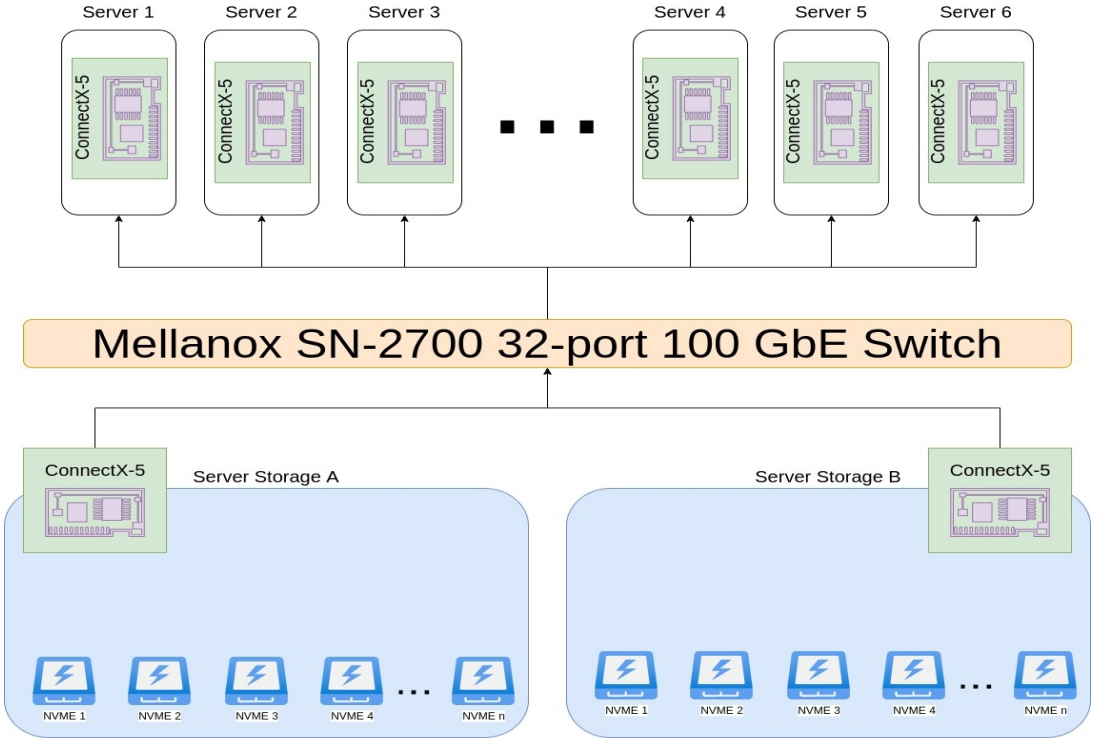
Low Latency

Extreme low latency is achieved by a combination of hardware offloading and accelerating mechanisms. As a result, end-to-end latency of RDMA sessions can be as low as 1000 nano-seconds or 1 micro-second.



RMDA Protocol

Suggestions for hardware and software

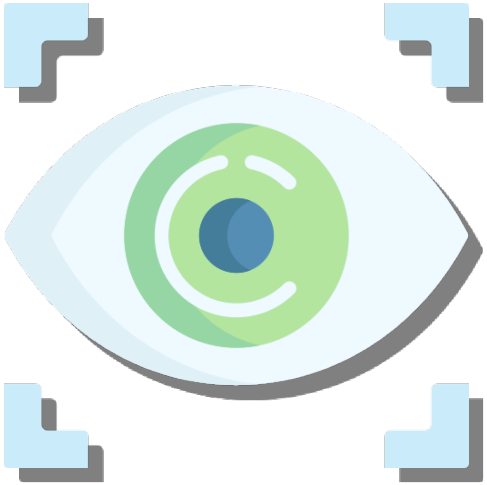


- **Hardware**
- Switch SN-2700 32-port 100 GbE/s
- NIC MCX516A-CCAT ConnectX-5
- CableMCP1600-C003E26N DAC (1to1)
- **Software**
- nfs over RDMA
- RDMA roce solutions
- Ethernet Drivers
- Firmware
- RDMA over converged Ethernet (RoCE)

Job Tracker

Real-time tracking of your sealing jobs

Track the duration of tasks completed by workers



Job tracking logs were saved to a log file

Check the mining log for task completion duration

Reference JobTracker to lotus code easily

Job Tracker

code

```
// jobTracker tracking job task time done (start, end, duration)
func jobTracker(hostname string, workerID string, sector string, taskType string, startTime time.Time) {
	var jobFile *os.File
	var err error
	endTime := time.Now()
	duration := endTime.Sub(startTime).String()
	job := fmt.Sprintf("\n%v\t%v\t%v\t%v\t%v\t%v\t%v", hostname, workerID[:8], sector, taskType, startTime.Format("2006-01-02 15:04:05"), endTime.Format("20

	// Get job log path from environment variable
	// if path not setted, set path to default ~/
	getLogPath := os.Getenv("LOTUS_MINER_JOB_LOG_PATH")
	if len(getLogPath) == 0 {
		getLogPath = "~/"
	}

	// Create job log file
	fileName := fmt.Sprintf("%v/jobs-%v.log", getLogPath, string(time.Now().Format("2006-01-02")))
	_, err = os.Stat(fileName)
	if os.IsNotExist(err) {
		if jobFile, err = os.Create(fileName); err != nil {
			logJob.Warnf("cannot create jobs log file, got error %v", err.Error())
		}
		if _, err = jobFile.WriteString("HostName Worker Sector TaskType StartTime EndTime Duration\n"); err != nil {
			logJob.Warnf("cannot write into job log file, got error %v", err.Error())
		}
	} else {
		if jobFile, err = os.OpenFile(fileName, os.O_APPEND|os.O_WRONLY, 0600); err != nil {
			logJob.Warnf("cannot open jobs log file, got error %v", err.Error())
		}
	}

	// Append job done in job log file
	if _, err = jobFile.WriteString(job); err != nil {
		logJob.Warnf("cannot write into job log file, got error %v", err.Error())
	}

	logJob.Infof("Worker %v in hostname %v taskType %v with duration %v done", workerID[:8], hostname, taskType, duration)

}
```

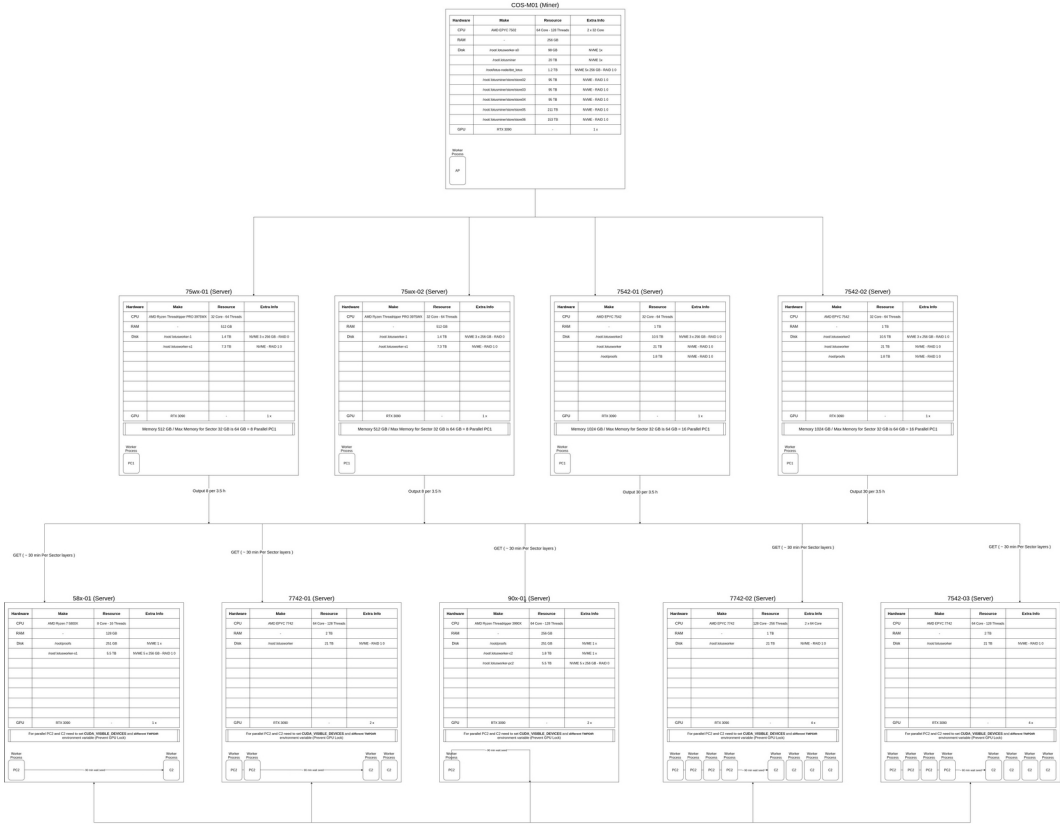
Job Tracker

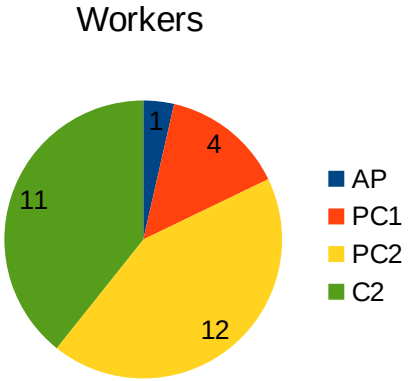
Result

```
2021-07-13T02:56:56.111-0400    INFO    jobTracker      sector-storage/worker_tracked.go:75     Worker 8a4acfbf in hostname ip-172-31-36-231
taskType PC1 with duration 9.51718ms done
```

HostName	Worker	Sector	TaskType	TimeStart	TimeEnd	Duration
ip-172-31-36-231	001c751d	2	AP	2021-07-30 03:18:20	2021-07-30 03:18:20	2.80551ms
ip-172-31-36-231	001c751d	2	AP	2021-07-30 03:18:20	2021-07-30 03:18:20	3.726633ms
ip-172-31-36-231	81346a03	2	GET	2021-07-30 03:27:04	2021-07-30 03:27:04	688.697µs
ip-172-31-36-231	001c751d	3	AP	2021-07-30 03:27:04	2021-07-30 03:27:04	5.427607ms
ip-172-31-36-231	001c751d	4	AP	2021-07-30 03:27:04	2021-07-30 03:27:04	4.000551ms
ip-172-31-36-231	81346a03	2	PC1	2021-07-30 03:27:04	2021-07-30 03:27:04	12.745323ms
ip-172-31-36-231	77a69335	2	GET	2021-07-30 03:27:04	2021-07-30 03:27:04	1.18404ms
ip-172-31-36-231	81346a03	4	GET	2021-07-30 03:27:04	2021-07-30 03:27:04	585.075µs
ip-172-31-36-231	81346a03	4	PC1	2021-07-30 03:27:04	2021-07-30 03:27:04	7.767657ms
ip-172-31-36-231	77a69335	4	GET	2021-07-30 03:27:04	2021-07-30 03:27:04	801.64µs
ip-172-31-36-231	77a69335	2	PC2	2021-07-30 03:27:04	2021-07-30 03:27:04	47.698506ms
ip-172-31-36-231	77a69335	4	PC2	2021-07-30 03:27:04	2021-07-30 03:27:04	8.953336ms
ip-172-31-36-231	001c751d	3	AP	2021-07-30 03:27:39	2021-07-30 03:27:39	2.961135ms
ip-172-31-36-231	81346a03	3	GET	2021-07-30 03:27:39	2021-07-30 03:27:39	546.214µs
ip-172-31-36-231	81346a03	3	PC1	2021-07-30 03:27:39	2021-07-30 03:27:39	6.398742ms
ip-172-31-36-231	77a69335	3	GET	2021-07-30 03:27:39	2021-07-30 03:27:39	692.088µs
ip-172-31-36-231	77a69335	3	PC2	2021-07-30 03:27:39	2021-07-30 03:27:39	7.871279ms
ip-172-31-36-231	001c751d	2	GET	2021-07-30 03:29:28	2021-07-30 03:29:28	95.413µs
ip-172-31-36-231	001c751d	4	GET	2021-07-30 03:29:28	2021-07-30 03:29:28	72.792µs
ip-172-31-36-231	001c751d	3	GET	2021-07-30 03:29:28	2021-07-30 03:29:28	101.433µs
ip-172-31-36-231	001c751d	2	C1	2021-07-30 03:29:28	2021-07-30 03:29:28	7.322445ms
ip-172-31-36-231	001c751d	4	C1	2021-07-30 03:29:28	2021-07-30 03:29:28	7.199132ms
ip-172-31-36-231	001c751d	3	C1	2021-07-30 03:29:28	2021-07-30 03:29:28	6.773522ms
ip-172-31-36-231	77a69335	2	C2	2021-07-30 03:29:28	2021-07-30 03:31:29	2m1.638826486s
ip-172-31-36-231	77a69335	3	C2	2021-07-30 03:29:28	2021-07-30 03:31:30	2m2.898961074s
ip-172-31-36-231	77a69335	4	C2	2021-07-30 03:29:28	2021-07-30 03:31:33	2m5.335449689s
ip-172-31-36-231	001c751d	3	GET	2021-07-30 03:32:11	2021-07-30 03:32:11	497.352µs
ip-172-31-36-231	001c751d	4	GET	2021-07-30 03:32:11	2021-07-30 03:32:11	106.743µs
ip-172-31-36-231	001c751d	3	FIN	2021-07-30 03:32:11	2021-07-30 03:32:11	283.798µs
ip-172-31-36-231	001c751d	2	GET	2021-07-30 03:32:11	2021-07-30 03:32:11	145.524µs
ip-172-31-36-231	001c751d	4	FIN	2021-07-30 03:32:11	2021-07-30 03:32:11	459.942µs
ip-172-31-36-231	001c751d	2	FIN	2021-07-30 03:32:11	2021-07-30 03:32:11	348.088µs

Lab servers


COS-M01 (Miner)
75wx-01 (Server)
75wx-02 (Server)
7542-01 (Server)
7542-02 (Server)
58x-01 (Server)
7742-01 (Server)
90x-01 (Server)
7742-02 (Server)
7542-03 (Server)


Workers
1
4
12
11
AP
PC1
PC2
C2