HyperSDK

Performance Optimization Guide

Maximize throughput with connection pooling, parallel downloads, streaming exports, and intelligent resource management for production workloads.

Connection Pooling -- Parallel I/O -- Bandwidth Throttling

Connection Pooling

Reuse authenticated connections to cloud provider APIs for 30-40% faster operations.

30-40%	5	60s	30s
Faster API Operations	Max Connections per Provider	Idle Connection Timeout	Health Check Interval

🔗 Connection Reuse

Authenticated sessions are pooled and reused across requests. Eliminates repeated TLS handshakes and authentication roundtrips that add 200-500ms per request.

🔧 Pool Configuration

Per-provider pool sizing with configurable max connections (1-20), idle timeout, and maximum lifetime. Defaults optimized for typical production workloads.

💚 Health Checks

Periodic health checks validate pooled connections. Stale or broken connections are automatically evicted and replaced, preventing request failures.

📈 Metrics & Monitoring

Pool utilization exposed via Prometheus metrics: active connections, idle connections, wait time, and eviction count for capacity tuning.

Configuration

```yaml
connection_pool:
  max_connections: 5        # Per-provider maximum
  idle_timeout: "60s"       # Close idle connections after
  max_lifetime: "300s"      # Maximum connection lifetime
  health_check: "30s"       # Health check interval
  wait_timeout: "10s"       # Max wait for available connection
```

Parallel Downloads

Configurable multi-worker downloads with chunk-based transfer and bandwidth throttling.

Worker Configuration

Configure 1-16 parallel download workers per job. Each worker handles an independent chunk of the disk image for maximum throughput on high-bandwidth links.

- **Default:** 4 workers
- **Range:** 1-16 configurable
- **Auto-tune:** Based on available bandwidth

Chunk Size Tuning

Adjustable chunk sizes from 1 MB to 64 MB. Larger chunks reduce overhead on fast networks; smaller chunks improve parallelism on high-latency connections.

- **LAN (1 Gbps+):** 32-64 MB chunks
- **WAN (100 Mbps):** 8-16 MB chunks
- **High latency:** 1-4 MB chunks

Bandwidth Throttling

Per-job bandwidth limits prevent export operations from saturating network links. Set limits in Mbps to protect production traffic during business hours.

- Per-job and global limits
- Time-based throttling profiles
- Burst allowance for short transfers

Resume & Retry

Automatic resume of interrupted downloads from the last completed chunk. No wasted bandwidth on transient failures. Configurable retry count with backoff.

- Chunk-level checksum verification
- Automatic retry with exponential backoff
- Progress persistence across restarts

Performance by Worker Count

Workers	50 GB VM (1 Gbps)	200 GB VM (1 Gbps)	CPU Overhead
1 (sequential)	7.5 min	30 min	Low (5%)
4 (default)	2.2 min	8.5 min	Moderate (15%)
8	1.3 min	5 min	Moderate (25%)
16	1.0 min	3.8 min	High (40%)

Export Optimization

OVA streaming, compression tuning, and format selection for optimal export performance.

OVA Streaming

Stream OVA exports directly to the target without writing intermediate files to local disk. Eliminates the need for 2x storage space and reduces total export time by 30-50%.

Compression Levels

Choose from no compression (fastest), fast (gzip level 1), balanced (level 6), or maximum (level 9). The right choice depends on network vs. CPU trade-offs.

Format Selection

Export format impacts performance significantly. VMDK thin is fastest for vSphere targets; QCOW2 offers best compression for KVM; raw for maximum compatibility.

Zero-Copy Transfers

Where supported, HyperSDK uses zero-copy (sendfile) transfers to move data between disk and network without user-space copies, maximizing I/O throughput.

Compression Impact on 100 GB VM Export

Level	Export Size	Export Time	CPU Usage	Best For
None	100 GB	15 min	5%	Fast LAN, local storage

Fast (level 1)	65 GB	18 min	20%	Balanced workloads
Balanced (level 6)	45 GB	25 min	50%	WAN transfers
Maximum (level 9)	40 GB	40 min	80%	Archival storage

> *For cross-region migrations over WAN links, balanced compression (level 6) typically gives the best end-to-end time by reducing network transfer at moderate CPU cost.*

Resource Management

Rate limiting, request size controls, memory management, and goroutine pools for stable operation.

Rate Limiting

Token-bucket rate limiter applied via the `HTTPMiddleware` port. Configurable requests per second with burst allowance. Protects both the daemon and upstream provider APIs.

- Default: 100 req/s, burst 20
- Per-client and global limits
- 429 responses with Retry-After header

Request Size Limits

Maximum request body size enforced at the middleware layer. Prevents memory exhaustion from oversized payloads. Configurable per-endpoint for upload vs. API routes.

- API requests: 1 MB default
- Upload endpoints: configurable
- 413 responses for oversized requests

Memory Management

Streaming I/O with fixed-size buffers prevents memory growth proportional to VM size. Memory usage stays constant regardless of export size. GC tuning for throughput.

- Fixed 32 MB I/O buffers
- Streaming (no full-file buffering)

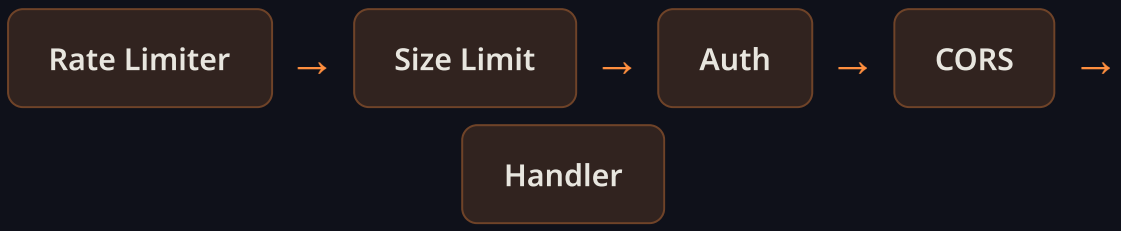
Goroutine Pools

Bounded worker pools for concurrent operations prevent goroutine explosion. Job execution, health checks, and cleanup tasks each use dedicated pools with configurable sizes.

- Job workers: 10 (default)
- Health check workers: 3
- Cleanup workers: 2

- GOGC tuned for throughput

Middleware Pipeline (Performance-Ordered)



Benchmarks

Real-world performance data: export times, network throughput, and CPU utilization across VM sizes.

Export Time by VM Size (4 Workers, Balanced Compression)

VM Size	vSphere Export	AWS Export	Azure Export	GCP Export
10 GB	1.5 min	2.0 min	2.2 min	1.8 min
50 GB	7 min	9 min	10 min	8 min
200 GB	28 min	35 min	38 min	32 min
1 TB	2.3 hrs	2.9 hrs	3.2 hrs	2.7 hrs

Resource Utilization During Export

850
Mbps Network Throughput

25%
Avg CPU Utilization

256
MB Peak Memory Usage

~50
Goroutines Under Load

API Response Times (p99)

Endpoint	p50	p95	p99
GET /api/v1/jobs	2 ms	8 ms	15 ms
POST /api/v1/jobs	5 ms	18 ms	35 ms
GET /api/v1/providers	3 ms	12 ms	25 ms
GET /api/v1/metrics	1 ms	4 ms	8 ms

Optimized for Production Workloads

HyperSDK delivers consistent, predictable performance from small VMs to multi-terabyte exports. Tune connection pools, worker counts, and compression to match your infrastructure.