HeuristicOptimizer

Documentacion de Administracion

Version 3.0.0

Paper 1.21.x . Folia 1.21.x . Java 21

Indice de Contenidos

Capitulo 1 — Instalacion y Primeros Pasos

1.1 Requisitos del Sistema

HeuristicOptimizer esta disenado exclusivamente para software de servidor Minecraft de nivel productivo. Las siguientes versiones minimas deben cumplirse antes de intentar la instalacion. Ejecutar el plugin en versiones no compatibles produce un fallo de arranque inmediato con un mensaje de error descriptivo en la consola.

Componente	Version Minima	Notas
Paper	1.21.11	Plataforma principal de soporte
Folia	1.21.11	Soporte nativo completo para Folia — no es una capa de compatibilidad
Java	21	Obligatorio. Java 17 no esta soportado.

1.2 Dependencias

1.2.1 Dependencia Obligatoria

ProtocolLib es una dependencia de ejecucion obligatoria. HeuristicOptimizer se negara a arrancar si ProtocolLib no esta presente en el directorio de plugins. Proporciona las primitivas de inspeccion y manipulacion de paquetes de red que utilizan varios submotores, incluyendo los de supervision de red y renderizado avanzado.

Plugin	Version Minima	Fuente
ProtocolLib	5.3.0+	spigotmc.org/resources/protocollib.1997/

Importante: ProtocolLib debe estar instalado e inicializado completamente antes de que HeuristicOptimizer se cargue. El orden de carga se gestiona automaticamente por Paper/Folia a traves de la directiva depend en plugin.yml.

1.2.2 Dependencias Opcionales

Los siguientes plugins se detectan en el arranque a traves del gestor de plugins del servidor. Cuando estan presentes, desbloquean capacidades de integracion adicionales. El motor de optimizacion principal funciona completamente sin ninguno de ellos.

Plugin	Capacidad Desbloqueada
LuckPerms	Caps de distancia de vision/simulacion por grupo; asignacion de inmunidad por permisos
PlaceholderAPI	Expone placeholders %ho_*% para plugins de TAB, scoreboards y displays holograficos
Spark	Integracion avanzada de profiling JVM y correlacion con eventos de optimizacion
WorldGuard	Flags de region para optimizacion: ho-exempt (omitir optimizacion) y ho-aggressive

1.3 Instalacion Paso a Paso

Paso 1 — Obtener el JAR del Plugin

Descarga `HeuristicOptimizer-3.0.0.jar` desde el canal de releases oficial. Utiliza unicamente builds oficiales. No renombres el archivo JAR; la cadena de version embebida en el nombre de archivo se lee durante el arranque para verificaciones internas de compatibilidad.

Paso 2 — Colocar los Archivos en plugins/

El siguiente esquema muestra la disposicion minima requerida de archivos. Los plugins opcionales pueden colocarse en cualquier orden junto a los obligatorios.

```
servidor/
  plugins/
    ProtocolLib.jar                  (obligatorio — debe estar presente)
    HeuristicOptimizer-3.0.0.jar
    LuckPerms.jar                    (opcional)
    PlaceholderAPI.jar               (opcional)
    WorldGuard.jar                   (opcional)
    Spark.jar                        (opcional)
```

Paso 3 — Primer Arranque del Servidor

Inicia el servidor normalmente. En la primera carga, HeuristicOptimizer genera su directorio de configuracion completo bajo `plugins/HeuristicOptimizer/`. No es necesario reiniciar inmediatamente; se aplican los valores predeterminados y el plugin comienza a operar en el modo configurado.

```
java -jar paper-1.21.4.jar
```

Paso 4 — Configurar el Idioma

Edita `bootstrap.yml` para establecer el idioma de la interfaz preferido antes del siguiente reinicio. Este archivo solo se lee durante el arranque del servidor.

```
# bootstrap.yml
language: es_ES      # Disponible: en_US, es_ES
log_level: INFO      # DEBUG | INFO | WARNING | SEVERE
```

Paso 5 — Reiniciar

```
/ho policyreload    # o reinicio completo del servidor
```

1.4 Estructura de Archivos Generada

El siguiente arbol de directorios se crea en el primer arranque. Cada archivo controla un aspecto distinto del sistema y tiene su propio mecanismo de recarga.

```
plugins/HeuristicOptimizer/
  config.yml              Configuracion principal — comportamiento, umbrales, parametros de handlers
  bootstrap.yml           Configuracion de arranque — idioma, nivel de log, rutas de archivos
  policies.hopl           Definiciones de politicas de optimizacion y selectores de objetivo
  constraints.hopl        Limites de seguridad de ejecucion y caps de tasa
  orchestration.hopl      Estrategia del orquestador, pesos de puntuacion, cooldowns
  observability.hopl      Intervalos de observacion del Eye y configuracion del modo Analysis
  lang/
    en_US.yml             Cadenas de idioma en ingles
    es_ES.yml             Cadenas de idioma en espanol
```

1.5 Configuracion Inicial

1.5.1 Procedimiento Recomendado para el Primer Arranque — Modo Analysis

Antes de habilitar optimizaciones automaticas en un entorno de produccion, se recomienda encarecidamente ejecutar el plugin en modo Analysis durante 30-60 minutos con carga de jugadores representativa. El sistema observara sin intervenir y generara un informe de configuracion adaptado a los patrones de carga reales del servidor.

Habilita el modo Analysis en `observability.hopl`:

```
# observability.hopl
analysis:
  enabled: true
  export-path: observations.log
  report-path: analysis-report.txt
```

Tras la ventana de observacion, desactiva el modo Analysis, reinicia el servidor y lee `analysis-report.txt` antes de ajustar los umbrales en `config.yml`.

1.6 Modos de Operacion

HeuristicOptimizer soporta cinco modos de operacion distintos. El modo activo determina el comportamiento del orquestador durante cada ciclo de evaluacion. Los modos pueden cambiarse en tiempo de ejecucion sin necesidad de reiniciar el servidor.

Modo	Comportamiento	Uso Recomendado
ACTIVE	Optimizacion automatica completa. El orquestador evalua las metricas en cada ciclo y aplica acciones segun las politicas activas.	Entornos de produccion estables
PASSIVE	Solo observacion e informes. Se recopilan y registran todas las metricas; no se ejecutan acciones.	Despliegue inicial, diagnostico
MANUAL	Acciones automaticas desactivadas. Solo /ho optimize y /ho trigger pueden aplicar cambios.	Escenarios de control total del administrador
DISABLED	Plugin completamente inactivo. Sin observacion, sin acciones, sin recopilacion de metricas.	Depuracion profunda y aislamiento de conflictos
ANALYSIS	La telemetria del Eye funciona a maxima resolucion. Los datos se exportan a disco. No se ejecutan acciones.	Profiling pre-produccion y captura de baseline

```
/ho mode active
/ho mode passive
/ho mode manual
/ho mode disabled
/ho mode analysis
```

1.7 Configuracion Rapida para Produccion

Los umbrales predeterminados en `config.yml` son intencionalmente bajos para garantizar actividad visible del plugin en servidores pequenos o de prueba. En un servidor de produccion estos valores provocaran intervenciones constantes e innecesarias. Ajusta a los siguientes rangos tras revisar el informe del modo Analysis.

```
# config.yml — seccion heuristics.triggers
heuristics:
  triggers:
    # Umbrales de entidades
    entity-count:        400     # Valor demo: 80
    passive-mob-count:   200     # Valor demo: 40
    entity-hotspot:      150     # Valor demo: 35  (por chunk)
    tile-entity-count:  20000    # Valor demo: 1000

    # Umbrales de chunks
    loaded-chunks:       8000    # Valor demo: 450
    simulated-chunks:    4000    # Valor demo: 220
    forced-chunks:        300    # Valor demo: 60
    chunk-entity-hotspot:  80    # Valor demo: 30

    # Redstone y mecanicas
    redstone-updates:   10000    # Valor demo: 500
    piston-activations:  2000    # Valor demo: 100

  engine:
    cooldown-min-seconds: 20     # Valor demo: 12
    cooldown-max-seconds: 300    # Valor demo: 180
```

> **Importante:** No ajustar estos umbrales antes de abrir un servidor de produccion a jugadores es la causa mas comun de intervenciones innecesarias en el TPS y quejas de experiencia de usuario. Ejecuta siempre el modo Analysis primero para establecer el rango de operacion normal de tu servidor.

1.8 Lista de Verificacion de Instalacion

- ProtocolLib aparece como cargado correctamente en el log de arranque sin errores
- `/ho status` devuelve "Eye: ACTIVO" dentro de los 10 segundos posteriores al arranque
- El conteo de submotores muestra 9/9 activos en la salida de status
- El modo de operacion esta configurado al valor deseado (ACTIVE para produccion)
- No aparecen errores ni stack traces de HeuristicOptimizer en el log de arranque
- Las integraciones opcionales (LuckPerms, PlaceholderAPI, WorldGuard, Spark) se detectan si estan instaladas
- Los umbrales de produccion se han aplicado a la seccion heuristics.triggers de config.yml
- El modo Analysis se ha ejecutado durante al menos 30 minutos y se ha revisado analysis-report.txt

1.9 Problemas Comunes de Instalacion

Error / Sintoma	Causa	Solucion
"ProtocolLib not found"	JAR de ProtocolLib ausente o con fallo de carga	Instala ProtocolLib 5.3.0+ y verifica arranque limpio
"Plugin failed to initialise"	Version de JRE inferior a Java 21	Actualiza JRE/JDK a Java 21
"API version mismatch"	Software de servidor inferior a 1.21.11	Actualiza Paper o Folia a 1.21.11+
"Eye not ready"	El Eye necesita aprox. 5 segundos de calentamiento	Espera 10 segundos tras el arranque y vuelve a ejecutar /ho status
TPS no mejora	Umbrales demo demasiado bajos para carga de produccion	Aplica umbrales de produccion segun la Seccion 1.7
Submotores muestran menos de 9/9	Un submotor encontro un error de inicializacion	Revisa la consola para el nombre especifico del submotor y el stack trace
"Failed to load policies.hopl"	Error de sintaxis YAML en un archivo .hopl	Ejecuta /ho policy test y corrige el error reportado

Capitulo 2 — Referencia de Comandos

Todos los comandos de HeuristicOptimizer se acceden a traves del comando base `/ho` (alias: `/heuristicoptimizer`). Cada subcomando requiere un nodo de permiso especifico bajo el espacio de nombres `heuristicoptimizer.*`. Los comandos funcionan tanto desde la consola del servidor como en el juego, salvo donde se indique explicitamente.

2.1 Sintaxis y Alias del Comando Base

```
/ho <subcomando> [argumentos...]
/heuristicoptimizer <subcomando> [argumentos...]

Permiso base requerido para cualquier acceso a /ho: heuristicoptimizer.use
```

2.2 Resumen de Subcomandos

Subcomando	Nodo de Permiso	Consola	Descripcion
/ho status	.inspect	Si	Muestra un snapshot completo del estado del sistema
/ho eye	.eye.view	No	Abre el menu GUI de telemetria en tiempo real
/ho scan	.inspect	Si	Escanea chunks en busca de hotspots de entidades
/ho watch	.inspect	Si	Monitorea a un jugador en tiempo real
/ho history	.history	Si	Muestra el historial de ejecucion de acciones
/ho policy	.policy.*	Si	Testear, revisar o recargar politicas
/ho optimize	.control	Si	Dispara una optimizacion inmediata para un jugador
/ho restore	.control	Si	Revierte todas las optimizaciones pendientes de un jugador
/ho profile	.control	Si	Listar o cambiar perfiles de optimizacion
/ho viewdist	.control	Si	Override de distancia de vision (jugador/mundo/region)
/ho simdist	.control	Si	Override de distancia de simulacion (misma sintaxis)
/ho dump	.report	Si	Exporta el estado completo del sistema a JSON
/ho simulate	.inspect	Si	Simulacion: muestra que haria el sistema sin ejecutar
/ho extensions	.inspect	Si	Lista las extensiones de API registradas
/ho trigger	.control	Si	Ejecuta un handler de accion especifico directamente

2.3 /ho status

Muestra un snapshot completo del estado del sistema en el momento de ejecucion. Todos los datos provienen del snapshot mas reciente del Eye y nunca tienen mas antiguedad que el intervalo de observacion configurado

para cada dominio.

```
/ho status
```

La salida incluye:

- Estado del Eye — activo/inactivo, timestamp del ultimo snapshot
- TPS actual — medias moviles de 1 minuto, 5 minutos y 15 minutos
- Total de chunks cargados y simulados en todos los mundos
- Conteo total de entidades en todos los chunks cargados
- Numero de jugadores conectados y lista de mundos cargados
- Conteo de submotores activos (ej. 9/9)
- Numero de politicas cargadas y activas
- Estadisticas de sesion: uptime, total de acciones ejecutadas / rechazadas / fallidas

2.4 /ho eye

Abre un menu GUI estilo inventario con telemetria del servidor en tiempo real, dividido en secciones por categoria. Este comando solo esta disponible para jugadores en el juego; no puede usarse desde la consola del servidor.

```
/ho eye
```

Secciones del GUI:

- **Proceso del Servidor** — duracion del tick, uptime, version, nivel de API
- **Memoria** — heap JVM usado/libre/maximo, uso off-heap, conteo de eventos GC recientes
- **Rendimiento** — TPS, MSPT, varianza del MSPT en los ultimos 60 segundos
- **Mundos** — por mundo: tipo de entorno, hora del dia, estado climatico, conteos de chunks
- **Distribucion de Chunks** — cargados, simulados, force-loaded, conteo de hotspots por mundo
- **Distribucion de Entidades** — total, desglose por EntityCategory, conteo de tile entities
- **Jugadores** — por jugador: mundo actual, VD efectiva, SD efectiva, ping
- **Red** — tasa de paquetes entrantes/salientes, ancho de banda estimado por segundo
- **Mecanicas** — activaciones de pistons/tick, transferencias de hoppers/tick, actualizaciones de redstone/tick

> **Nota:** Hacer clic en el icono de cualquier categoria fuerza una actualizacion inmediata de los datos de ese dominio, saltando el intervalo de observacion normal. Requiere `heuristicoptimizer.eye.forceupdate`.

2.5 /ho scan

Escanea todos los chunks dentro de un radio dado e informa aquellos cuyo conteo de entidades o tile-entities supera los umbrales de advertencia configurados. Los resultados se ordenan por severidad e incluyen enlaces de teleporte clicables cuando se usa en el juego.

```
/ho scan [mundo] [radio]

Ejemplos:
  /ho scan              # Mundo actual, radio 5 (por defecto)
  /ho scan world 10     # Mundo "world", radio 10 chunks
  /ho scan nether 3     # Mundo Nether, radio 3 chunks
```

Parametro	Tipo	Por Defecto	Descripcion
mundo	string	Mundo del jugador que ejecuta	Nombre del mundo objetivo
radio	entero	5	Radio de escaneo en chunks desde la posicion del jugador

Los umbrales de advertencia (`commands.scan.entity-warn-threshold`, `commands.scan.tile-entity-warn-threshold`) y el numero maximo de resultados (`commands.scan.max-results`) son configurables en `config.yml`.

2.6 /ho watch

Monitorea a un jugador conectado en tiempo real durante una duracion configurable (por defecto 10 segundos). Se emite una linea de monitoreo por ciclo de evaluacion del servidor, registrando el contexto actual del jugador. Util para correlacionar la posicion y actividad de un jugador con caidas de TPS observadas.

```
/ho watch <jugador>

Ejemplo de salida (una linea por segundo):
  [t:84521] TPS:19.2 Ping:72ms VD:10 SD:8 Pos:312,64,-820 (world)
  [t:84541] TPS:18.7 Ping:68ms VD:10 SD:8 Pos:312,64,-820 (world)
```

> **Nota:** La duracion de observacion es configurable mediante `commands.watch.duration-seconds` en config.yml.

2.7 /ho history

Recupera las entradas mas recientes del registro de ejecucion de acciones. Cada entrada registra el contexto completo del servidor en el momento de ejecucion, incluyendo la justificacion de la decision y el resultado. Es la herramienta principal para entender lo que el orquestador ha estado haciendo en el servidor.

```
/ho history [limite]

  limite: numero de registros (1-100, por defecto: 10)

Ejemplo de salida:
  [2024-11-15 20:14:33] PAUSE_MOB_AI — SUCCESS
    Razon: entity-hotspot superado (312 > 150) en world en chunk (18,-52)
    Condiciones: cond[TPS:15.2 MSPT:64ms E:1820 P:47]
```

2.8 /ho policy

Operaciones sobre las politicas de optimizacion cargadas. El subcomando `reload` es el mecanismo principal de recarga en caliente para todos los archivos de configuracion `.hopl` y no requiere reiniciar el servidor.

```
/ho policy test      # Validar archivos .hopl en busca de errores de sintaxis (sin aplicar cambios)
/ho policy review    # Mostrar todas las politicas cargadas con sus parametros
/ho policy reload    # Recargar en caliente policies.hopl, constraints.hopl,
                     # orchestration.hopl y observability.hopl
```

Subcomando	Permiso	Descripcion
test	.policy.test	Parsea todos los archivos .hopl; reporta errores sin aplicar ningun cambio
review	.policy.review	Lista politicas activas: ID, capability, selector, estado habilitado y todos los parametros
reload	.policy.reload	Recarga atomicamente todos los archivos .hopl; las politicas en ejecucion continuan hasta que la recarga se completa

2.9 /ho optimize

Dispara un ciclo de optimizacion inmediato para el jugador especificado, saltando el intervalo de evaluacion normal y los temporizadores de cooldown. El parametro de modo opcional sobreescribe el perfil activo solo para esta ejecucion.

```
/ho optimize <jugador> [modo]

  modo (opcional): aggressive | moderate | conservative

Ejemplos:
  /ho optimize Steve               # Usa el perfil activo actualmente
  /ho optimize Steve aggressive    # Fuerza el perfil agresivo para este ciclo
```

2.10 /ho restore

Fuerza la ejecucion inmediata de todos los auto-reverts pendientes para el jugador especificado. Normalmente los reverts se disparan automaticamente cuando se cumplen las condiciones de recuperacion. Este comando los fuerza independientemente del estado actual del servidor. Usalo cuando un jugador reporta una distancia de vision persistentemente reducida despues de que un evento de lag ha sido resuelto completamente.

```
/ho restore <jugador>
```

2.11 /ho profile

Lista todos los perfiles de optimizacion disponibles o cambia el perfil activo. Los cambios de perfil tienen efecto inmediato y persisten hasta el proximo cambio o reinicio del servidor.

```
/ho profile list             # Mostrar todos los perfiles con sus umbrales
/ho profile set <id>         # Cambiar el perfil activo inmediatamente

IDs disponibles: survival | minigames | creative | hardcore | aggressive | conservative
```

2.12 /ho viewdist

Control manual fino sobre la distancia de vision de los jugadores. Los overrides creados por este comando tienen mayor prioridad que cualquier politica automatica. Los overrides pueden aplicarse a un jugador individual, a un mundo completo o a una region rectangular de chunks dentro de un mundo.

```
/ho viewdist set player  <n> <distancia>
/ho viewdist set world   <mundo> <distancia>
/ho viewdist set region  <mundo> here <radio_chunks> <distancia>
/ho viewdist set region  <mundo> <x1> <z1> <x2> <z2> <distancia>

/ho viewdist clear player  <n>
/ho viewdist clear world   <mundo>
/ho viewdist clear region  <mundo> <x1> <z1> <x2> <z2>

/ho viewdist status <jugador>
/ho viewdist status world <mundo>

Rango de distancia valido: 2-32 chunks
```

> **Nota:** Un override de alcance PLAYER tiene la prioridad mas alta posible en el sistema, por encima de los caps de permiso, las regiones de WorldGuard y los perfiles activos. Solo puede ser reemplazado por otro override de alcance PLAYER.

2.13 /ho simdist

Sintaxis identica a `/ho viewdist` pero controla la distancia de simulacion. La distancia de simulacion nunca puede superar la distancia de vision; cualquier override que resultara en SD > VD se clampea automaticamente a la VD efectiva actual.

```
/ho simdist set player  <n> <distancia>
/ho simdist set world   <mundo> <distancia>
/ho simdist set region  <mundo> here <radio> <distancia>
/ho simdist clear player <n>
/ho simdist status <jugador>

Restriccion: SD <= VD siempre aplicado. Valor minimo: 2 chunks.
```

2.14 /ho dump

Exporta un snapshot completo del estado interno del plugin a un archivo JSON con timestamp. Pensado para compartir con soporte tecnico o para analisis offline.

```
/ho dump
# Salida: plugins/HeuristicOptimizer/dumps/dump-<ISO-timestamp>.json
```

Contenido del JSON:

- Cadena de version del plugin y perfil activo actualmente
- Todos los auto-reverts pendientes con sus condiciones de disparo y timestamps de expiracion
- Estadisticas del historial de TPS: minimo, maximo, media y desviacion estandar de la sesion
- Snapshot completo del Eye en el momento del dump: TPS, MSPT, memoria, entidades, chunks, datos por jugador
- Registros de efectividad de acciones: tasa de exito por tipo de accion y delta de TPS medio

2.15 /ho simulate

Realiza una simulacion completa del ciclo de orquestacion para el contexto actual del jugador especificado sin ejecutar ninguna accion. Genera el plan de decision completo: que acciones se seleccionarian, su puntuacion de prioridad compuesta, el ID de capability y la justificacion. Esencial para verificar que la configuracion de politicas se comporta como se espera antes de aplicarla en produccion.

```
/ho simulate <jugador>

Ejemplo de salida:
  [PLAN] Jugador: Steve  |  Perfil: survival
  [1] PAUSE_MOB_AI       prioridad: 0.87  cap: entities   razon: hotspot@(18,-52)
  [2] SET_VIEW_DISTANCE  prioridad: 0.64  cap: rendering  razon: TPS 15.2 < 17.0
  [3] THROTTLE_REDSTONE  prioridad: 0.41  cap: redstone   razon: updates 8420 > 8000
```

2.16 /ho extensions

Lista todos los submotores y proveedores de metricas registrados mediante la API publica de Java. Muestra el nombre de la extension, el tipo (ExternalSubmotor o MetricProvider), el timestamp de registro y el estado activo actual dentro del ciclo de orquestacion.

```
/ho extensions list
```

2.17 /ho trigger — Ejecucion Directa de Handlers

Ejecuta un handler de accion especifico directamente, saltando el orquestador, todos los cooldowns, circuit breakers y la evaluacion de politicas. Los parametros se suministran como pares `clave=valor` y se parsean automaticamente al tipo Java correcto (boolean, int, double o String). Los siguientes valores de contexto se inyectan automaticamente desde la posicion del jugador ejecutor: `worldName`, `chunkX`, `chunkZ`, `playerUUID`.

Este comando esta pensado para pruebas empiricas de configuracion e intervenciones manuales de emergencia, no para administracion rutinaria del servidor.

```
/ho trigger <tipo_accion> [clave=valor ...]

# Pausar IA de mobs en un radio de 6 chunks
/ho trigger pause_mob_ai pause=true radius=6

# Reanudar IA de mobs
/ho trigger pause_mob_ai pause=false radius=6

# Throttle de hoppers — 8 ticks entre cada transferencia
/ho trigger throttle_hoppers ticksPerTransfer=8

# Smart hopper throttle — max 4 transferencias por segundo
/ho trigger smart_hopper_throttle enable=true maxTransfersPerSec=4

# Reducir densidad de particulas en un 80% para jugadores cercanos
/ho trigger reduce_particle_density reduction=80

# Activar despawn heuristico de items cuando MSPT supere 40ms
/ho trigger heuristic_item_despawn enabled=true msptThreshold=40.0

# Limitar spawns de monstruos a 20
/ho trigger set_spawn_limit category=MONSTER spawnLimit=20

# Throttle de redstone a maximo 20 cambios de estado por segundo
/ho trigger throttle_redstone_updates updatesPerSecond=20

# Aplicar preset agresivo de rango de seguimiento de entidades
/ho trigger set_tracking_range preset=aggressive

# Forzar distancia de vision a 6 chunks para jugadores en rango
/ho trigger set_view_distance targetViewDistance=6
```

Nota: La inferencia de tipos en los parametros es automatica. `true`/`false` se convierten en boolean, los valores numericos con decimal en double, los enteros en int y cualquier otro valor se trata como String. No se requieren anotaciones de tipo.

Capitulo 3 — Archivos de Configuracion

HeuristicOptimizer utiliza seis archivos de configuracion principales, cada uno con un alcance distinto y un comportamiento de recarga propio. Entender que archivo gobierna cada sistema es esencial para un ajuste eficaz en produccion.

Archivo	Proposito	Recarga en Caliente
bootstrap.yml	Seleccion de idioma, nivel de log, rutas de archivos .hopl	No — requiere reinicio
config.yml	Todo el comportamiento en tiempo de ejecucion: umbrales, handlers, logica de auto-revert	Parcial — ver nota
policies.hopl	Definiciones de politicas de optimizacion y declaraciones de selectores	Si — /ho policy reload
constraints.hopl	Limites de seguridad de ejecucion, caps de tasa, guardas de uso de recursos	Si — /ho policy reload
orchestration.hopl	Estrategia del orquestador, pesos de puntuacion de prioridad, cooldowns	Si — /ho policy reload
observability.hopl	Intervalos de muestreo de dominios del Eye y configuracion del modo Analysis	Si — /ho policy reload

> **Nota:** Los cambios en secciones de config.yml distintas a `heuristics.triggers` requieren un reinicio completo del servidor para tener efecto. La seccion `heuristics.triggers` se recarga en caliente mediante `/ho policy reload`.

3.1 bootstrap.yml

Controla la configuracion exclusiva del arranque. Se lee una sola vez durante la inicializacion del plugin; se ignora durante el resto de la sesion del servidor. Todos los cambios requieren un reinicio completo.

```
# bootstrap.yml

# Idioma de la interfaz
# Valores disponibles: en_US, es_ES
language: es_ES

# Nivel de verbosidad del log
# DEBUG   — todas las decisiones internas, evaluaciones de metricas y salida de ciclos
# INFO    — salida operacional estandar (recomendado para produccion)
# WARNING — solo advertencias y errores
# SEVERE  — solo errores criticos
log_level: INFO

# Rutas a los archivos .hopl (relativas a plugins/HeuristicOptimizer/)
paths:
  policies:      policies.hopl
  constraints:   constraints.hopl
  orchestration: orchestration.hopl
  observability: observability.hopl
```

3.2 config.yml

3.2.1 auto-regulation

Cambio automatico de perfil segun un horario diario. Usa formato de 24 horas en la zona horaria local de la JVM del servidor.

```
auto-regulation:
  enabled: false          # Establecer true para activar el cambio programado
  profile: survival       # Perfil activo al arrancar el servidor
  schedule:
    - time: "08:00"
      profile: survival
    - time: "20:00"
      profile: aggressive   # Anticipar la carga del horario pico
    - time: "03:00"
      profile: conservative # Nocturno — intervencion minima
  notify-admins: true     # Notificar el cambio de perfil a los admins online
```

3.2.2 discord

Notificaciones por webhook de Discord. Los fallos de entrega se registran pero no afectan al funcionamiento del plugin.

```
discord:
  webhook-url: ""                    # Cadena vacia desactiva las notificaciones
  alert-tps-threshold: 14.0         # La alerta se activa cuando el TPS baja de este valor
  notify-automatic-actions: true     # Publicar mensaje por cada accion automatica ejecutada
  notify-profile-changes: true       # Publicar mensaje al cambiar de perfil
```

3.2.3 handlers — Configuracion por Accion

Parametros predeterminados para cada handler de accion. Estos valores se utilizan cuando el orquestador selecciona una accion sin un override de parametros explicito proveniente de una politica.

```
handlers:
  pause-mob-ai:
    default-radius: 6           # Radio de chunks por defecto para la suspension de IA
    max-radius: 12              # Radio maximo (no puede ser superado por las politicas)
    exclude-pets: true          # Nunca pausar la IA de animales domesticados
    exclude-villagers: false    # Establecer true para proteger siempre a los aldeanos
    exclude-villagers-threshold: 0
    auto-renew: true            # Reaplicar automaticamente si las condiciones persisten tras el
revert

  aggressive-item-merge:
    default-chunk-radius: 6
    max-chunk-radius: 12

  throttle-hoppers:
    default-ticks: 5            # Ticks entre transferencias (vanilla es 8)
    min-ticks: 1
    max-ticks: 1200
    default-radius: 0           # 0 = global (todos los chunks del mundo)
    max-radius: 12
    default-max-transfers-per-sec: 4

  throttle-redstone:
    default-updates-per-second: 20
    max-updates-per-second: 200

  set-spawn-limit:
    default-limit: 50
    default-category: "MONSTER"  # MONSTER | ANIMAL | WATER_ANIMAL | AMBIENT

  entity-tracking:
    default-distance: 8
```

3.2.4 auto-revert

Controla cuando y como se revierten las optimizaciones automaticas una vez que la condicion desencadenante se ha resuelto. Soporta senales de recuperacion tanto globales del servidor como regionalmente localizadas.

```
auto-revert:
  enabled: true

  metrics:
    tps-recovery-threshold-pct: 30.0    # % de mejora en TPS desde el baseline al momento de la
accion
    mspt-recovery-threshold-pct: 20.0
    entity-recovery-threshold-pct: 20.0
    player-recovery-threshold-pct: 10.0
    chunk-recovery-threshold-pct: 10.0
    # false = cualquier metrica individual que cumpla su umbral dispara el revert
    # true  = todas las metricas configuradas deben cumplir sus umbrales simultaneamente
    require-all-metrics: false

  regional:
    enabled: true
    action-types:                    # Solo estos tipos usan evaluacion de revert regional
      - pause-mob-ai
      - set-view-distance
      - set-simulation-distance
    require-all-metrics: true
    required-consecutive-confirmations: 2   # Ciclos que deben confirmar la recuperacion
    signal:
      mode: composite                  # composite | hotspots-only
      weights:
        entity-hotspots: 0.60
        chunk-hotspots: 0.25
        players: 0.15

  min-uptime-before-revert-ms:
    default: 120000                    # 2 minutos de tiempo activo minimo
    pause-mob-ai: 120000
    set-view-distance: 180000          # 3 minutos minimo
    set-simulation-distance: 180000
```

3.2.5 distance-overrides

```
distance-overrides:
  queue:
    reset-after-single-apply: true    # Aplicar un override y luego limpiar la cola (previene
cascadas)

  dynamic:
    enabled: true                    # Activar degradacion proporcional segun la carga del servidor
    target-tps: 20.0
    target-mspt: 50.0
    multiplier-min: 0.50             # El cap de VD puede reducirse hasta el 50% del cap de permiso
    multiplier-max: 1.00
    min-delta-chunks: 1              # Cambio minimo para considerarlo significativo
    cooldown-ms: 15000               # Minimo 15s entre ajustes dinamicos
    actionbar:
      enabled: true
      min-interval-ms: 10000
      template: "[HO] Vision limitada: {current}/{max} chunks"
```

3.2.6 engine

```
engine:
  min-evaluation-interval-ms: 10000    # Tiempo minimo entre ciclos completos de orquestacion
  history-max-records: 2000
  history-max-file-size-mb: 10
  history-persist-interval-ms: 5000   # Volcar el historial de acciones a disco cada 5 segundos
```

3.2.7 heuristics — Motor de Activacion Principal

La seccion mas critica de config.yml. Define cada umbral que determina cuando el orquestador considera necesaria una intervencion, y los parametros adaptativos que gobiernan la intensidad de la respuesta.

```
heuristics:
  engine:
    tps-baseline-floor: 18.0          # Suelo EMA — evita que el baseline colapse bajo carga
sostenida
    tps-critical: 14.0                 # Umbral de severidad CRITICAL (maxima intensidad de
respuesta)
    tps-high: 17.0                     # Umbral de severidad HIGH

    # El cooldown escala linealmente: min al 100% de efectividad, max al 0%
    cooldown-min-seconds: 12
    cooldown-max-seconds: 180

    circuit-breaker-failures: 3        # Fallos consecutivos para abrir el circuit breaker
    circuit-breaker-silence-seconds: 120

    aimd:
      additive-increase: 0.05          # Incremento del factor por ciclo de accion exitoso
      multiplicative-decrease: 0.5     # Reduccion del factor en caso de fallo
      min-factor: 0.1
      max-factor: 1.0
      initial-factor: 0.5

  triggers:
    # Entidades
    entity-count: 80                   # DEMO — produccion: 300-500
    passive-mob-count: 40              # DEMO — produccion: 150-300
    entity-hotspot: 35                 # DEMO — produccion: 100-200 (por chunk)
    tile-entity-count: 1000            # DEMO — produccion: 10000-50000

    # Chunks
    loaded-chunks: 450                 # DEMO — produccion: 5000-20000
    simulated-chunks: 220              # DEMO — produccion: 2000-10000
    forced-chunks: 60                  # DEMO — produccion: 200-500
    chunk-entity-hotspot: 30           # DEMO — produccion: 50-100

    # Redstone y mecanicas
    redstone-updates: 500              # DEMO — produccion: 5000-20000
    piston-activations: 100            # DEMO — produccion: 1000-5000
```

> **Importante:** Todos los valores marcados como DEMO son intencionalmente bajos para entornos de prueba. Para produccion, multiplica los umbrales de entidades y chunks por 5-10x y los de redstone/pistons por 10-20x. Valida siempre con el modo Analysis primero.

3.3 policies.hopl

3.3.1 Selectores

Los selectores definen grupos logicos de chunks o jugadores. Multiples politicas pueden referenciar el mismo selector. Los selectores se evaluan de forma lazy en el momento de ejecucion de la politica.

```
selectors:
  - id: hot_chunks
    type: chunk
    criteria:
      loaded: true
      has_players: true
      entity_count_min: 50

  - id: all_chunks
    type: chunk
    criteria:
      loaded: true

  - id: afk_players
    type: player
    criteria:
      ticks_since_last_move_min: 1200    # 60 segundos sin moverse

  - id: high_ping_players
    type: player
    criteria:
      ping_min: 200
```

3.3.2 Politicas

Una politica vincula una capability (tipo de accion) a un selector, con parametros de ejecucion opcionales. El orquestador puntua todas las politicas habilitadas en cada ciclo y selecciona las mejores propuestas.

```
policies:
  - id: entity_cleanup_aggressive
    capability: remove_entities
    selector: hot_chunks
    enabled: true
    max_entities_per_chunk: 150
    entity_types: [DROPPED_ITEM, ARROW, EXPERIENCE_ORB]

  - id: redstone_throttle_heavy
    capability: throttle_redstone
    selector: all_chunks
    enabled: true
    max_updates_per_tick: 500

  - id: mob_ai_pause_hotspots
    capability: pause_mob_ai
    selector: hot_chunks
    enabled: true
    radius: 6
    exclude_pets: true
```

3.4 constraints.hopl

```
execution:
  max_actions_per_cycle: 10
  max_actions_per_minute: 50
  max_actions_per_hour: 500
  cooldown_seconds: 15

safety:
  require_confirmation: false
  allow_partial_execution: true
  rollback_on_error: true

resource_limits:
  max_memory_usage_percent: 90
  min_tps_for_execution: 8.0
  max_mspt_for_execution: 90.0
```

3.5 orchestration.hopl

```
strategy: priority_based

priority_scoring:
  impact_weight: 0.45
  safety_weight: 0.35
  efficiency_weight: 0.20

submotor_priority:
  PlayerQoS: 1
  Entities: 2
  ChunkSpace: 3
  Redstone: 4
  WorldSimulation: 5
  Rendering: 6
  GameMechanics: 7
  Network: 8

cooldowns:
  global_cooldown_seconds: 15
  per_submotor_cooldown_seconds: 30
  per_capability_cooldown_seconds: 60
```

3.6 observability.hopl

```
eye:
  observation:
    # El intervalo se expresa en ticks del servidor (20 ticks = 1 segundo)
    server:    { enabled: true,  interval: 20  }    # Cada 1 segundo
    worlds:    { enabled: true,  interval: 60  }    # Cada 3 segundos
    chunks:    { enabled: true,  interval: 40  }    # Cada 2 segundos
    entities:  { enabled: true,  interval: 60  }    # Cada 3 segundos
    players:   { enabled: true,  interval: 20  }    # Cada 1 segundo
    network:   { enabled: true,  interval: 20  }    # Cada 1 segundo
    mechanics: { enabled: true,  interval: 100 }    # Cada 5 segundos
    plugins:   { enabled: false, interval: 72000 }  # Cada 1 hora (costoso)
    advanced:  { enabled: false, interval: 40  }    # Requiere ProtocolLib

analysis:
  enabled: false
  export-path: observations.log
  report-path: analysis-report.txt

# Opciones de tracking por dominio (ejemplo para el dominio mechanics)
mechanics:
  track-redstone: true
  track-pistons: true
  track-hoppers: true
  track-random-ticks: true
  track-mob-spawning: true
  track-combat-events: false      # Sobrecarga muy alta — solo para diagnostico
  track-physics-events: false     # Sobrecarga muy alta — solo para diagnostico
```

3.7 Archivos de Idioma

Todas las cadenas de texto visibles al usuario estan externalizadas en archivos de idioma bajo `lang/`. Se incluyen dos idiomas: `en_US.yml` (ingles) y `es_ES.yml` (espanol). El idioma activo se selecciona en `bootstrap.yml`. Los archivos de idioma utilizan codigos de color de Minecraft (`&6`, `&a`, `&c`, etc.) y placeholders con nombre como `{tps}`, `{playerName}` y `{action}`. Se pueden crear traducciones personalizadas copiando un archivo existente y modificando los valores de las cadenas.

Capitulo 4 — Perfiles de Optimizacion y Permisos

4.1 Perfiles de Optimizacion Predefinidos

Un perfil es un preset con nombre que ajusta los umbrales de TPS a los que se activa cada categoria de optimizacion principal, y que tipos de accion estan permitidos dentro de ese perfil. Los perfiles permiten cambios rapidos de agresividad en todo el servidor sin necesidad de editar valores individuales de umbral.

ID de Perfil	Caso de Uso Ideal	Agresividad
survival	Servidores survival clasicos con comportamiento y granjas estandar	Media
minigames	Servidores de juego con arenas, multiples entidades temporales y resets	Alta
creative	Servidores creativos donde cualquier interrupcion del juego es inaceptable	Baja
hardcore	Servidores hardcore donde la estabilidad del servidor es primordial	Alta
aggressive	Maximo rendimiento a cualquier coste — emergencias o situaciones de estres	Maxima
conservative	Intervencion minima — observar mas que actuar	Minima

4.2 Detalle de Umbrales por Perfil

La siguiente tabla muestra el valor de TPS al que se activa cada categoria de optimizacion principal para cada perfil. Un valor de 17.0 significa que la accion se dispara cuando el TPS cae por debajo de 17.0.

Categoria de Activacion	survival	minigames	creative	hardcore	aggressive	conservative
Reduccion de VD	17.0	18.0	19.0	16.0	19.0	15.0
Pausa de IA de Mobs	15.0	17.0	19.0	14.0	18.0	13.0
Limite de Spawns	16.0	17.0	19.0	15.0	18.0	14.0
Merge Agresivo de Items	No	Si	No	Si	Si	No
Priorizar Drop de XP	Si	No	Si	Si	No	Si

4.3 Cambio de Perfil

```
/ho profile list             # Ver todos los perfiles e identificar el activo
/ho profile set aggressive   # Cambiar a aggressive inmediatamente
/ho profile set survival     # Volver a survival
```

4.4 Regulacion Automatica por Horario

Cuando `auto-regulation.enabled: true`, el plugin cambia perfiles automaticamente segun el horario diario configurado. Especialmente efectivo para servidores con patrones de carga pico y valle consistentes.

```
auto-regulation:
  enabled: true
  schedule:
    - time: "08:00"     # Manana — carga estandar
      profile: survival
    - time: "20:00"     # Pico vespertino — maximizar rendimiento
      profile: aggressive
    - time: "03:00"     # Nocturno — pocos jugadores, conservar recursos
      profile: conservative
```

4.5 Nodos de Permiso — Acceso a Comandos

Todos los nodos de permiso utilizan el prefijo `heuristicoptimizer.`. Por defecto todos los permisos funcionales requieren OP. Los permisos para no-OP deben asignarse explicitamente a traves de un gestor de permisos como LuckPerms.

Nodo de Permiso	Otorga Acceso A	Por Defecto
heuristicoptimizer.use	Acceso base al /ho	OP
heuristicoptimizer.inspect	status, scan, watch, simulate, extensions	OP
heuristicoptimizer.eye.view	/ho eye	OP
heuristicoptimizer.eye.forceupdate	Forzar actualizacion en el GUI del Eye	OP
heuristicoptimizer.history	/ho history	OP
heuristicoptimizer.control	optimize, restore, profile, viewdist, simdist, trigger	OP
heuristicoptimizer.report	/ho dump	OP
heuristicoptimizer.audit	Acceso al log de auditoria	OP
heuristicoptimizer.api	Acceso a la API publica de Java	OP
heuristicoptimizer.*	Todos los permisos	OP

4.6 Nodos de Permiso — Operaciones de Politicas

Nodo de Permiso	Descripcion
heuristicoptimizer.policy.reload	Recargar todos los archivos .hopl mediante /ho policy reload
heuristicoptimizer.policy.test	Validar sintaxis .hopl mediante /ho policy test — sin aplicar cambios
heuristicoptimizer.policy.review	Inspeccionar politicas cargadas mediante /ho policy review

4.7 Permiso de Inmunidad

Un jugador que posea `heuristicoptimizer.immune` queda excluido completamente de todas las acciones de optimizacion automatica. Su distancia de vision no sera reducida, los chunks a su alrededor no seran objetivo de pausas de IA ni merges de entidades, y no aparecera como fuente desencadenante en ninguna evaluacion heuristica. Este permiso debe otorgarse a todo el personal del servidor y administradores tecnicos.

```
# Ejemplo con LuckPerms
/lp group staff permission set heuristicoptimizer.immune true
```

> **Nota:** Los operadores (OP) eluden automaticamente todos los caps de permiso. El permiso de inmunidad solo es necesario para miembros del staff no-OP que necesiten proteccion frente a las acciones de optimizacion.

4.8 Caps de Distancia por Permiso

4.8.1 Caps de Distancia de Vision

El nodo de permiso de cap de VD establece la distancia de vision maxima que un jugador puede recibir de cualquier fuente — politica automatica, degradacion dinamica u override manual del administrador en P6 o inferior. El plugin nunca asignara una VD superior a este cap al jugador.

```
# Formato estandar (recomendado)
heuristicoptimizer.viewdistance.max.<N>

# Formatos legacy (soportados por compatibilidad retroactiva)
heuristicoptimizer.viewdistance.<N>
heuristicoptimizer.vd.<N>

# N debe ser un entero de 2 a 32

# Ejemplos de asignacion con LuckPerms
/lp group default permission set heuristicoptimizer.viewdistance.max.6
/lp group vip     permission set heuristicoptimizer.viewdistance.max.12
/lp group premium permission set heuristicoptimizer.viewdistance.max.20
```

4.8.2 Caps de Distancia de Simulacion

```
# Formato estandar (recomendado)
heuristicoptimizer.simdistance.max.<N>

# Formato legacy
heuristicoptimizer.sd.<N>

/lp group default permission set heuristicoptimizer.simdistance.max.4
/lp group vip     permission set heuristicoptimizer.simdistance.max.8
/lp group premium permission set heuristicoptimizer.simdistance.max.12
```

> **Nota:** La distancia de simulacion nunca puede superar la distancia de vision. Si un cap de SD resultara en SD > VD, se clampea automaticamente a la VD efectiva actual.

4.9 Integracion con LuckPerms — Modelo de Monetizacion de Referencia

El sistema de caps por permiso esta disenado especificamente para soportar modelos de monetizacion por niveles. El siguiente es un ejemplo de implementacion de referencia para una red con cuatro niveles.

```
# Nivel gratuito — experiencia estandar
/lp group free    permission set heuristicoptimizer.viewdistance.max.6
/lp group free    permission set heuristicoptimizer.simdistance.max.4

# Nivel VIP
/lp group vip     permission set heuristicoptimizer.viewdistance.max.12
/lp group vip     permission set heuristicoptimizer.simdistance.max.8

# Nivel Premium
/lp group premium permission set heuristicoptimizer.viewdistance.max.20
/lp group premium permission set heuristicoptimizer.simdistance.max.12

# Staff — inmune a toda optimizacion automatica
/lp group staff   permission set heuristicoptimizer.immune true
```

4.10 Degradacion Dinamica

Cuando `distance-overrides.dynamic.enabled: true`, el cap efectivo de cada jugador se calcula continuamente como proporcion de su cap de permiso, escalado segun la carga actual del servidor. La ventaja relativa entre niveles se mantiene siempre.

```
Calculo del multiplicador:
  tps_ratio  = tps_actual / tps_objetivo
  mspt_ratio = mspt_objetivo / mspt_actual
  multiplicador = max(multiplicador_min, min(1.0, min(tps_ratio, mspt_ratio)))
  cap_efectivo  = floor(cap_permiso * multiplicador)

Ejemplo practico:
  Cap de permiso:  12 chunks (grupo VIP)
  TPS actual:      15.0  (objetivo: 20.0) -> tps_ratio  = 0.75
  MSPT actual:     62ms  (objetivo: 50ms) -> mspt_ratio = 0.81
  Multiplicador:   max(0.50, min(1.0, 0.75)) = 0.75
  Cap efectivo:    floor(12 * 0.75) = 9 chunks

  Notificacion al jugador via Action Bar:
  "[HO] Vision limitada: 9/12 chunks"
```

4.11 Jerarquia Completa de Prioridad

Cuando multiples fuentes intentan establecer la distancia de vision o simulacion de un jugador, el siguiente orden de prioridad determina el resultado. Un numero P mayor siempre gana.

```
P8 (maxima) Comando manual del admin          /ho viewdist set player Steve 10
P7          Permiso de inmunidad               heuristicoptimizer.immune
P6          Cap de distancia por permiso       heuristicoptimizer.viewdistance.max.12
P5          Flag de region WorldGuard          ho-exempt o ho-aggressive en la region
P4          Override a nivel de mundo          config.yml heuristics.worlds.<mundo>
P3          Override a nivel de jugador        config.yml heuristics.players.<uuid>
P2          Perfil de optimizacion activo      /ho profile set aggressive
P1 (minima) Reglas heuristicas base            config.yml heuristics.triggers.*
```

Ejemplos de resolucion de conflictos:

• `/ho viewdist set player Steve 10` — Steve recibe VD=10. Este override P8 prevalece sobre todo, incluyendo caps, perfiles y regiones.

• Steve posee `.immune` — no se le aplica ninguna optimizacion automatica en absoluto, independientemente del estado del servidor. (P7)

• Steve tiene `.viewdistance.max.12` y la politica solicita VD=6 — Steve recibe VD=6. El cap no se excede, P6 no interviene.

• Steve tiene `.viewdistance.max.8` y la politica solicita VD=10 — Steve recibe VD=8. La solicitud se clampea al cap. (P6 gana sobre P2)

• Steve esta dentro de una region con flag `ho-exempt` — no se aplican optimizaciones en esa region. (P5)

4.12 Flags de Region WorldGuard

Flag	Valores	Comportamiento
ho-exempt	allow / deny	allow: todas las acciones de HeuristicOptimizer se omiten para los jugadores dentro de esta region. Usar en zonas de spawn, mercados y regiones de infraestructura critica de jugadores.
ho-aggressive	allow / deny	allow: se aplica siempre el conjunto de acciones mas agresivo disponible en esta region, independientemente del perfil global activo. Usar en arenas PvP y zonas de juego de alta densidad.

```
/rg flag spawn   ho-exempt     allow   # Proteger spawn de toda optimizacion
/rg flag arena   ho-aggressive allow   # Maximizar siempre la optimizacion en arenas
/rg flag mercado ho-exempt     allow   # Proteger tiendas y mercados de jugadores
```

Capitulo 5 — Guia de Operacion

5.1 Ciclo de Vida e Interno del Plugin

El plugin opera en un ciclo interno continuo gobernado por `engine.min-evaluation-interval-ms`. Cada ciclo pasa por las siguientes fases. Comprender este flujo ayuda a interpretar la salida del historial de acciones y diagnosticar comportamientos inesperados.

Fase	Descripcion
Observacion	El Eye sondea todos los dominios del servidor habilitados — entidades, chunks, jugadores, mecanicas, red y otros — y construye un snapshot del estado del servidor en ese instante. Cada dominio se actualiza segun su intervalo de muestreo configurado.
Analisis	Nueve submotores independientes evaluan cada uno el snapshot contra su dominio. Los submotores cubren areas como gestion de entidades, presion de chunks, carga de redstone, calidad de servicio de jugadores y salud de la red. Cada uno produce cero o mas propuestas de accion priorizadas.
Orquestacion	Todas las propuestas se puntuan, deduplicanfiltran contra restricciones de seguridad, temporizadores de cooldown, estado del circuit breaker y limites de uso de recursos. Las propuestas con mayor puntuacion que superan todos los filtros se seleccionan para ejecucion.
Ejecucion	Las acciones seleccionadas se despachan al executor de plataforma correspondiente. Paper y Folia tienen rutas de ejecucion separadas para garantizar el contexto de hilo correcto en cada plataforma.
Retroalimentacion	Los resultados se registran en el historial de acciones. Las metricas de efectividad y los parametros de control adaptativo se actualizan. Se inician temporizadores de auto-revert para las acciones reversibles.

> **Nota:** El algoritmo de puntuacion especifico, los limites de dominio de cada submotor y los mecanismos internos de retroalimentacion son propietarios. Esta descripcion es suficiente para el diagnostico operacional y el ajuste de la configuracion.

5.2 Que Hace el Plugin Automaticamente en Modo ACTIVE

Accion	Condicion de Activacion	Efecto Visible para el Jugador
Reducir VD y SD	Caida de TPS combinada con alto numero de chunks cargados	Los jugadores ven una distancia de render menor
Pausar IA de mobs	Alto conteo de entidades o hotspot por chunk detectado	Los mobs cercanos dejan de moverse temporalmente
Throttle de redstone	Tasa de actualizaciones de redstone por encima del umbral	Los circuitos de redstone responden mas lentamente
Reducir densidad de particulas	TPS por debajo del umbral de particulas	Se renderizan menos particulas en el cliente
Limitar spawns de mobs	Conteo total de entidades por encima del umbral	Nacen menos mobs nuevos durante el periodo
Fusionar items caidos	Alto conteo de items en uno o mas chunks	Los stacks de items se consolidan en menos entidades

Accion	Condicion de Activacion	Efecto Visible para el Jugador
Despawn heuristico de items	MSPT por encima del umbral configurado	Los items sueltos desaparecen antes que en vanilla
Congelar entidades distantes	Chunks sin jugadores cercanos	Las entidades en areas vacias pausan IA y movimiento
Optimizar pathfinding de mobs	Gran numero agregado de mobs	La evaluacion de goals de mobs usa logica simplificada

5.3 Que No Hace el Plugin Nunca

- Matar jugadores ni reducir directamente su salud o hambre
- Eliminar o alterar bloques colocados ni estructuras del mundo
- Modificar permanentemente archivos del mundo, archivos de region o level.dat
- Cambiar permanentemente las reglas del juego del servidor (las gamerules pueden ajustarse temporalmente como maximo)
- Transmitir analiticas o telemetria sin que `analytics.enabled: true` este configurado explicitamente por el administrador
- Modificar archivos de configuracion ni datos en tiempo de ejecucion de otros plugins instalados

5.4 Escenario: Servidor Survival (50+ Jugadores)

Problema: El TPS cae regularmente a 14-15 durante el horario pico vespertino debido a la operacion simultanea de granjas de mobs, redstone activo e instalaciones, y altas distancias de vision por jugador en un mundo ampliamente explorado. La intervencion manual a esta escala es insostenible.

```
# config.yml — configuracion de produccion recomendada para survival
auto-regulation:
  enabled: true
  schedule:
    - time: "06:00"
      profile: survival
    - time: "18:00"
      profile: aggressive    # Anticipar la carga pico
    - time: "01:00"
      profile: conservative  # Nocturno

heuristics:
  triggers:
    entity-count: 400
    passive-mob-count: 200
    entity-hotspot: 150
    loaded-chunks: 10000
    redstone-updates: 8000
    piston-activations: 1500
```

Resultado esperado: Durante el horario pico con perfil agresivo, el plugin reduce preventivamente VD/SD para jugadores del nivel gratuito, throttlea los circuitos de redstone que superen el umbral y pausa la IA de mobs en los chunks hotspot. Al remitir la carga pico, el auto-revert restaura la configuracion previa incrementalmente a medida que se confirman las condiciones de recuperacion.

5.5 Escenario: Servidor de Minijuegos

Problema: Las arenas de juego generan grandes cantidades temporales de entidades e items durante las sesiones activas. Entre sesiones la carga es normal. El sistema debe ser reactivo pero no debe interferir con el lobby ni los periodos entre partidas.

```
# config.yml
heuristics:
  triggers:
    entity-count: 200
    entity-hotspot: 80

# WorldGuard — proteger lobby, intensificar tratamiento de arenas
/rg flag lobby ho-exempt     allow
/rg flag arena ho-aggressive allow

# Establecer el perfil base apropiado
/ho profile set minigames
```

5.6 Escenario: Red con Rangos VIP

Problema: El servidor ofrece rangos VIP y Premium que prometen una experiencia de juego sostenida superior. Durante carga alta, todos los jugadores se ven igualmente afectados, erosionando la propuesta de valor de los rangos de pago.

```
# Asignacion de permisos con LuckPerms
/lp group free    permission set heuristicoptimizer.viewdistance.max.6
/lp group free    permission set heuristicoptimizer.simdistance.max.4
/lp group vip     permission set heuristicoptimizer.viewdistance.max.12
/lp group vip     permission set heuristicoptimizer.simdistance.max.8
/lp group premium permission set heuristicoptimizer.viewdistance.max.20
/lp group premium permission set heuristicoptimizer.simdistance.max.12
/lp group staff   permission set heuristicoptimizer.immune true

# Activar degradacion proporcional en config.yml
distance-overrides:
  dynamic:
    enabled: true
    target-tps: 20.0
    multiplier-min: 0.50
```

Resultado: Bajo carga sostenida, todos los niveles se degradan proporcionalmente. Un jugador gratuito puede ver su VD reducida de 6 a 3 chunks. Un VIP ve 12 reducidos a 6. Premium ve 20 reducidos a 10. La brecha relativa entre niveles se mantiene siempre. La propuesta de valor de cada rango se preserva incluso en las peores condiciones de carga.

5.7 Escenario: Diagnostico de Lag Inexplicable

Problema: El servidor experimenta caidas periodicas de TPS sin una causa unica obvia. Las herramientas de profiling estandar muestran que el tick es lento pero no aislan el sistema especifico responsable.

Paso 1 — Activar el modo Analysis

```
# observability.hopl
analysis:
  enabled: true
```

Reiniciar. Permitir 30-60 minutos de actividad de jugadores representativa con carga normal.

Paso 2 — Desactivar el modo Analysis y leer el informe

```
# observability.hopl
analysis:
  enabled: false

# Revisar el informe generado:
plugins/HeuristicOptimizer/analysis-report.txt
```

Paso 3 — Investigacion manual

```
/ho scan world 10              # Identificar chunks hotspot por radio
/ho watch JugadorSospechoso    # Monitorear al jugador cerca de la fuente sospechada
/ho simulate JugadorSospechoso # Ver que acciones tomaria el sistema ahora mismo
/ho eye                        # Abrir el GUI de telemetria en vivo
```

Paso 4 — Aplicar intervencion dirigida

```
/ho optimize JugadorSospechoso aggressive   # Forzar un ciclo de optimizacion agresivo
/ho trigger pause_mob_ai pause=true radius=8  # Pausa de IA manual si es necesario
/ho history 20                               # Verificar acciones ejecutadas y su efecto
```

5.8 Flujo de Trabajo Diario

Verificacion matutina

```
/ho status              # Confirmar Eye activo, los 9 submotores funcionando
/ho history 10          # Revisar acciones automaticas nocturnas
/ho profile list        # Confirmar que el perfil esperado esta activo
```

Durante el horario pico

```
/ho eye                 # Telemetria en vivo — monitorear TPS, MSPT, conteos de entidades
/ho status              # Verificacion rapida del estado en texto
/ho scan world 5        # Comprobacion puntual de nuevos hotspots de entidades
```

Si el TPS cae inesperadamente

```
/ho scan world 10                   # Identificar chunks fuente
/ho watch <jugador>                  # Monitorear jugador cercano al hotspot
/ho history 5                        # Confirmar que las acciones automaticas se dispararon
/ho optimize <jugador> aggressive    # Forzar intervencion agresiva si es necesario
```

Tras un incidente

```
/ho history 50                    # Revision completa de las acciones del incidente
/ho dump                           # Exportar snapshot completo para analisis offline
/ho restore <jugador>              # Revertir si un jugador reporta VD degradada persistente
```

5.9 Intervencion Manual con /ho trigger

Todos los handlers de accion son accesibles directamente mediante `/ho trigger` para uso de emergencia y pruebas empiricas. El comando salta el orquestador, todos los cooldowns y toda la evaluacion de politicas.

```
# Pausar toda la IA de mobs en un radio de emergencia de 10 chunks
/ho trigger pause_mob_ai pause=true radius=10

# Reanudar IA de mobs
/ho trigger pause_mob_ai pause=false radius=10

# Throttle duro de todo el redstone a 5 actualizaciones/seg
/ho trigger throttle_redstone_updates updatesPerSecond=5

# Ralentizar significativamente todos los hoppers
/ho trigger throttle_hoppers ticksPerTransfer=40 radius=0

# Eliminar la mayoria de particulas para jugadores cercanos
/ho trigger reduce_particle_density reduction=90

# Cap de spawn de emergencia
/ho trigger set_spawn_limit category=MONSTER spawnLimit=5

# Rango de seguimiento de entidades agresivo
/ho trigger set_tracking_range preset=aggressive

# Forzar distancia de vision para pruebas
/ho trigger set_view_distance targetViewDistance=4
```

5.10 Guia de Solucion de Problemas

El plugin no esta optimizando nada

- Ejecuta `/ho status` — confirma que el modo es ACTIVE, no PASSIVE, MANUAL o DISABLED
- Verifica que los umbrales en `config.yml` son apropiados para los conteos de entidades y chunks del servidor — los valores demo son demasiado bajos
- Comprueba que el TPS este realmente por debajo de los umbrales configurados; si el TPS es 19.5 y el umbral es 17.0, ninguna accion esta justificada
- Ejecuta `/ho simulate <jugador>` para confirmar que los submotores estan generando propuestas
- Ejecuta `/ho policy review` para verificar que las politicas estan cargadas con los parametros correctos

El TPS no mejora aunque el plugin este activo

- Ejecuta `/ho scan` — verifica que el sistema esta apuntando a los chunks hotspot reales y no a los adyacentes
- Revisa `/ho history` — confirma que las acciones estan alcanzando el estado SUCCESS y no REJECTED o ERROR

• Si las acciones tienen exito pero el TPS no mejora, la causa raiz puede estar fuera del alcance de HeuristicOptimizer (por ejemplo, sobrecarga del tick de otros plugins)

• Aumenta la agresividad: `/ho profile set aggressive`

• Ejecuta `/ho dump` y revisa los registros de efectividad para identificar que tipos de accion tienen bajo delta de TPS

Un jugador se queja de distancia de vision reducida

• Ejecuta `/ho viewdist status <jugador>` para identificar que nivel de prioridad establecio la VD actual

• Si la fuente es una politica automatica: el auto-revert la restaurara cuando las condiciones se recuperen — informa al jugador y monitorea

• Para alivio inmediato: `/ho viewdist set player <n> <distancia>` (override P8)

• Para exencion permanente: otorga `heuristicoptimizer.immune` o sube el cap `.viewdistance.max.*` de su grupo

El auto-revert no restaura la configuracion

• Confirma que `auto-revert.enabled: true` esta en config.yml

• Verifica que haya transcurrido `min-uptime-before-revert-ms` — el minimo predeterminado es 2-3 minutos

• Comprueba que los umbrales de recuperacion no esten configurados de forma irrealista para el rango de TPS normal del servidor

• Ejecuta `/ho history` e identifica los eventos de revert — aparecen como NO_OP o REJECTED?

Errores en consola durante el arranque

Mensaje de Error	Causa	Solucion
"ProtocolLib not found"	ProtocolLib ausente o con fallo de carga	Instala ProtocolLib 5.3.0+; verifica arranque limpio
"Failed to load policies.hopl"	Error de sintaxis YAML en archivo .hopl	Ejecuta /ho policy test; corrige la linea reportada
"Java version not supported"	Version de JRE inferior a Java 21	Actualiza JRE/JDK a Java 21
"API version mismatch"	Paper o Folia inferior a 1.21.11	Actualiza el software del servidor
"Sub-engine failed to init"	Error de inicializacion de un submotor	Revisa la consola para el nombre del submotor y el stack trace

5.11 Integracion con Discord

Cuando se proporciona una URL de webhook de Discord en config.yml, el plugin publica notificaciones estructuradas para eventos operacionales clave. Los fallos de entrega se registran pero no afectan al funcionamiento del plugin.

Eventos que disparan notificaciones:

• El TPS cae por debajo de `discord.alert-tps-threshold` (por defecto: 14.0)

• Se ejecuta una accion de optimizacion automatica (si `notify-automatic-actions: true`)

• El perfil activo cambia — ya sea manualmente o por el horario de auto-regulation

```
discord:
  webhook-url: "https://discord.com/api/webhooks/TU_ID/TU_TOKEN"
  alert-tps-threshold: 14.0
  notify-automatic-actions: true
  notify-profile-changes: true
```

5.12 Variables de PlaceholderAPI

Cuando PlaceholderAPI esta instalado, los siguientes placeholders estan disponibles para plugins de TAB, scoreboards, displays holograficos y cualquier otro plugin compatible con PAPI.

Placeholder	Devuelve	Valor de Ejemplo
%ho_tps%	TPS actual (media movil de 1 minuto)	19.8
%ho_mspt%	MSPT medio actual	45.2
%ho_mode%	Cadena del modo de operacion activo	ACTIVE
%ho_profile%	Identificador del perfil activo	survival
%ho_entities%	Conteo total de entidades en todos los mundos	1523
%ho_chunks%	Conteo total de chunks cargados en todos los mundos	4521

5.13 Archivos Generados en Tiempo de Ejecucion

Los siguientes archivos se escriben o actualizan durante la operacion normal del servidor. No deben editarse manualmente mientras el servidor este en ejecucion.

Archivo	Contenido	Notas
action_history.json	Registro completo de ejecucion de acciones con condiciones y resultados	Limitado por history-max-records en config.yml
action_effectiveness.json	Tasa de exito por tipo de accion y delta de TPS medio	Usado por el factor AIMD y el escalado de cooldowns
view_distance_overrides.yml	Todos los overrides de VD activos indexados por alcance y objetivo	Persiste entre reinicios del servidor
simulation_distance_overrides.yml	Todos los overrides de SD activos indexados por alcance y objetivo	Persiste entre reinicios del servidor
observations.log	Datos de observacion brutos del Eye en formato estructurado	Solo se escribe cuando el modo Analysis esta habilitado
analysis-report.txt	Diagnostico de lag legible y recomendaciones de configuracion	Generado al final de cada sesion de Analysis
dumps/*.json	Snapshots completos del estado del sistema generados por /ho dump	Con timestamp; no se limpian automaticamente