

You must now upload using a token.

Upload with either the global upload token or the repo upload token. Admins can manage the [global upload token settings](#).

Dismiss

forest651l / godl / pulls / 7

feat: implement comprehensive bandwidth throttling system

Open

4 minutes ago forest651l authored [#7](#) [CI Passed](#) [feature/bandwidth-throttling](#)

HEAD a4025bf	Patch	Change
81.19%	69.30%	+0.03%

Source

Coverage data is based on HEAD [a4025bf](#) (1 uploads) compared to BASE [4568142](#) (2 uploads)

Commits have different number of coverage report uploads [learn more](#)

















Files changed<sup>8</sup> Indirect changes<sup>8</sup> Commits<sup>3</sup> Flags<sup>1</sup> Components<sup>0</sup> File e

Uncovered  Partial  Covered

All flags

Name	Missed lines	Head %	Patch %	Change %
internal/concurrent/manager.go	30	78.26%	8.57%	-10.22%
<div><div>-11,6 +11,8</div><div><div>1111"time"</div><div>1212</div><div>1313"github.com/forest6511/godl/pkg/progressMgr"1414+ "github.com/forest6511/godl/pkg/ratelimit.Limiter"1515+ "github.com/forest6511/godl/pkg/types"1616)type ConcurrentDownloadManager struct {</div></div><div><div>-18,6 +20,7</div><div><div>1820chunker *Chunker</div><div>1921progressMgr *progress.Manager</div><div>2022wg sync.WaitGroup</div><div>2123+ ratelimiter ratelimit.Limiter</div><div>2224}</div><div>2225</div></div></div><div><div>View full file</div></div></div>				

23	26	// NewConcurrentDownloadManager creates a ne
-27,6 +30,20		<a href="#">View full file</a>
27	30	}
28	31	}
29	32	
	33	+ // NewConcurrentDownloadManagerWithOptions c
	34	+ func NewConcurrentDownloadManagerWithOptions
	35	+ manager := &ConcurrentDownloadManager+
	36	+ progressMgr: progress.NewManag
	37	+ }
	38	+
	39	+ // Create rate limiter if MaxRate is s
	40	+ if options != nil && options.MaxRate >
	41	+ manager.rateLimiter = ratelimit
	42	+ }
	43	+
	44	+ return manager
	45	+ }
	46	+
30	47	// Download performs concurrent download of
31	48	func (m *ConcurrentDownloadManager) Download
32	49	// Get file size first
-72,6 +89,7		<a href="#">View full file</a>
72	89	m.workers[i].ChunkInfo = chunk
73	90	m.workers[i].Progress = progre
74	91	m.workers[i].Error = errorChar
	92	+ m.workers[i].RateLimiter = m.r
75	93	}
76	94	
77	95	// Start workers
-165,6 +183,13		<a href="#">View full file</a>
165	183	for {
166	184	n, err := resp.Body.Read(buffe
167	185	if n > 0 {
	186	+ // Apply rate limiting
	187	+ if w.RateLimiter != n
	188	+ if rateLimiter
	189	+ return
	190	+ }
	191	+ }
	192	+ }
! 168	! 193	if _, writeErr := file
! 169	! 194	return fmt.Err
! 170	! 195	}
-344,6 +369,31		<a href="#">View full file</a>
344	369	}
345	370	defer func() { _ = file.Close() }()
346	371	
347		- _, err = io.Copy(file, resp.Body)
	372	+ // Copy with rate limiting if enabled
	373	+ if m.rateLimiter != nil {
	374	+ buffer := make([]byte, 32*1024
	375	+ }

		375	+	<code>for {</code>
		376	+	
		377	+	<code>n, readErr := resp.Body</code>
		378	+	<code>if n &gt; 0 {</code>
		379	+	<code>// Apply rate</code>
		380	+	<code>if rateLimiter</code>
		381	+	<code>return</code>
		382	+	<code>}</code>
		383	+	<code>if _, writeErr</code>
		384	+	<code>return</code>
		385	+	<code>}</code>
		386	+	<code>}</code>
		387	+	
		388	+	<code>if readErr == io.EOF {</code>
		389	+	<code>break</code>
		390	+	<code>}</code>
		391	+	<code>if readErr != nil {</code>
		392	+	<code>return fmt.Err</code>
		393	+	<code>}</code>
		394	+	<code>}</code>
		395	+	<code>return nil</code>
		396	+	<code>}</code>
		398	+	<code>_, err = io.Copy(file, resp.Body)</code>
349		399		<code>return err</code>
350		400	}	

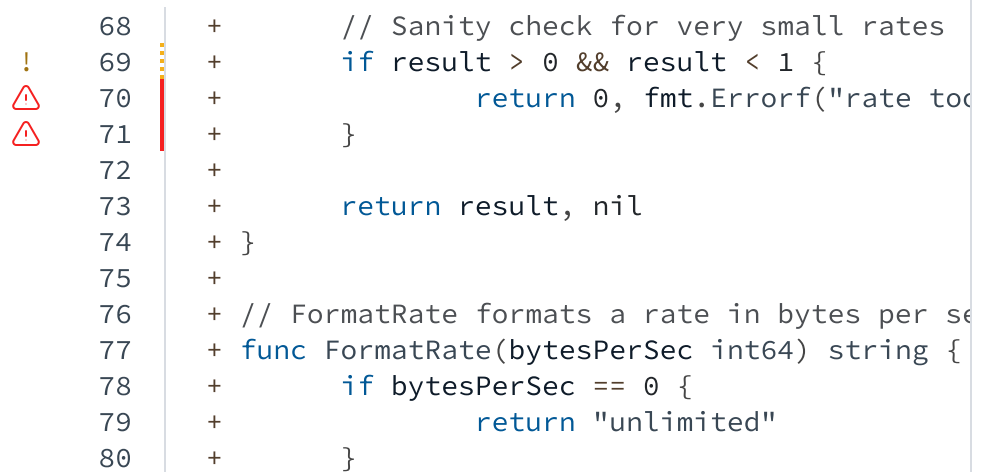
▼ pkg/ratelimit/parser.go

9 84.00% 84.00%

-

-0,0 +1,113

[View full file](#) 



```
68 + // Sanity check for very small rates
69 + if result > 0 && result < 1 {
70 +     return 0, fmt.Errorf("rate too
71 + }
72 +
73 +     return result, nil
74 + }
75 +
76 + // FormatRate formats a rate in bytes per se
77 + func FormatRate(bytesPerSec int64) string {
78 +     if bytesPerSec == 0 {
79 +         return "unlimited"
80 +     }
```

	81	+	
	82	+	const (
	83	+	KB = 1024
	84	+	MB = KB * 1024
	85	+	GB = MB * 1024
	86	+	)
	87	+	
	88	+	switch {
	89	+	case bytesPerSec >= GB:
	90	+	if bytesPerSec%GB == 0 {
	91	+	return fmt.Sprintf("%c
	92	+	}
⚠	93	+	return fmt.Sprintf("%.1fGB/s",
	94	+	case bytesPerSec >= MB:
	95	+	if bytesPerSec%MB == 0 {
	96	+	return fmt.Sprintf("%c
	97	+	}
	98	+	return fmt.Sprintf("%.1fMB/s",
	99	+	case bytesPerSec >= KB:
	100	+	if bytesPerSec%KB == 0 {
	101	+	return fmt.Sprintf("%c
	102	+	}
	103	+	return fmt.Sprintf("%.1fKB/s",
	104	+	default:
	105	+	return fmt.Sprintf("%d bytes/s
	106	+	}
	107	+	}
	108	+	
	109	+	// ValidateRate checks if a rate string is v
	110	+	func ValidateRate(rateStr string) error {
	111	+	_, err := ParseRate(rateStr)
	112	+	return err
	113	+	}

internal/core/downloader.go

7 81.06% 33.33% -0.79%

-21,6 +21,7		<a href="#">View full file</a>	
21	21		"github.com/forest6511/godl/internal/s
22	22		"github.com/forest6511/godl/pkg/errors
23	23		"github.com/forest6511/godl/pkg/progre
	24	+	"github.com/forest6511/godl/pkg/ratel-
24	25		"github.com/forest6511/godl/pkg/types"
25	26		)
26	27		
-1253,6 +1254,14		<a href="#">View full file</a>	
1253	1254		) (int64, error) {
1254	1255		buffer := make([]byte, options.ChunkS-
1255	1256		
	1257	+	// Create rate limiter if max rate is
	1258	+	var rateLimiter ratelimit.Limiter
!	1259	+	if options.MaxRate > 0 {
⚠	1260	+	rateLimiter = ratelimit.NewBar
!	1261	+	} else {
	1262	+	rateLimiter = ratelimit.NewNuT

	1263	+	}
	1264	+	
1256	1265		<code>var totalBytes int64</code>
1257	1266		
1258	1267		<code>lastProgressUpdate := time.Now()</code>
-1272,6 +1281,15			<a href="#">View full file</a>
1272	1281		<code>// Read chunk</code>
1273	1282		<code>n, err := src.Read(buffer)</code>
1274	1283		<code>if n &gt; 0 {</code>
	1284	+	<code>// Apply rate limiting</code>
	1285	+	<code>if rateLimiterErr := r</code>
	1286	+	<code>return totalBy</code>
	1287	+	<code>rateL</code>
	1288	+	<code>errors</code>
	1289	+	<code>"Downl</code>
	1290	+	<code>)</code>
	1291	+	<code>}</code>
	1292	+	
1276	1294		<code>written, writeErr := c</code>
1277	1295		<code>if writeErr != nil {</code>
1278	1296		<code>return totalBy</code>

▼ cmd/godl/main.go

6 67.47% 66.67% -0.05%

-20,6 +20,7			<a href="#">View full file</a>
20	20		<code>"github.com/forest6511/godl/internal/s</code>
21	21		<code>"github.com/forest6511/godl/pkg/cli"</code>
22	22		<code>"github.com/forest6511/godl/pkg/plugin</code>
	23	+	<code>"github.com/forest6511/godl/pkg/ratel</code>
23	24		<code>"github.com/forest6511/godl/pkg/types"</code>
24	25		<code>"github.com/forest6511/godl/pkg/ui"</code>
25	26		<code>)</code>
-60,6 +61,7			<a href="#">View full file</a>
60	61		<code>proxy string</code>
61	62		<code>output_format string</code>
62	63		<code>continuePartial bool</code>
	64	+	<code>maxRate string // Maximum d</code>
63	65		<code>// Plugin-related configurations</code>
64	66		<code>plugins []string</code>
65	67		<code>storageURL string</code>
-704,6 +706,16			<a href="#">View full file</a>
704	706		<code>}</code>
705	707		<code>}</code>
706	708		
	709	+	<code>// Configure max rate if specified</code>
	710	+	<code>if cfg.maxRate != "" {</code>
	711	+	<code>if maxRateBytes, err := ratel</code>
	712	+	<code>options.MaxRate = maxF</code>
	713	+	<code>} else {</code>
	714	+	<code>// Note: Error handlin</code>
	715	+	<code>fmt.Fprintf(os.Stderr,</code>
	716	+	<code>}</code>
	717	+	<code>}</code>

707	718	+	-
708	719		<code>return options</code>
709	720	}	
709	721		
-869,6 +881,12		<a href="#">View full file</a>	
869	881		<code>"Add custom header (can be use</code>
870	882		<code>)</code>
871	883		<code>flag.Var(&amp;headerFlags, "H", "Add custo</code>
	884	+	<code>flag.StringVar(</code>
	885	+	<code>&amp;cfg.maxRate,</code>
	886	+	<code>"max-rate",</code>
	887	+	<code>"",</code>
	888	+	<code>"Maximum download rate (e.g.,</code>
	889	+	<code>)</code>
872	890		
873	891		<code>// Initialize headers map and plugins</code>
874	892		<code>cfg.headers = make(<code>map</code>[string]string)</code>
-891,6 +909,13		<a href="#">View full file</a>	
891	909		<code>cfg.plugins = append(cfg.plugins,</code>
892	910		<code>)</code>
893	911		
	912	+	<code>// Validate max-rate if specified</code>
	913	+	<code>if cfg.maxRate != "" {</code>
	914	+	<code>    if err := ratelimit.ValidateRa</code>
	915	+	<code>        return nil, "", fmt.Er</code>
	916	+	<code>    }</code>
	917	+	<code>}</code>
	918	+	
894	919		<code>// Handle -c as an alias for --concurr</code>
895	920		<code>cWasSet := false</code>
896	921		
-1413,6 +1438,8		<a href="#">View full file</a>	
1413	1438		<code>--concurrent N      Number of concurre</code>
1414	1439		<code>--chunk-size SIZE    Chunk size <code>for</code> cor</code>
1415	1440		<code>Examples: 1MB, 512</code>
	1441	+	<code>--max-rate RATE      Maximum download r</code>
	1442	+	<code>Examples: 1MB/s, 5</code>
1416	1443		<code>--no-concurrent      Force single-threa</code>
1417	1444		<code>--no-color            Disable colored ou</code>
1418	1445		<code>--interactive        Enable interactive</code>
-1440,6 +1467,7		<a href="#">View full file</a>	
1440	1467		<code>%s https://example.com/file.zip</code>
1441	1468		<code>%s --concurrent 8 https://example.com/larg</code>
1442	1469		<code>%s --chunk-size 2MB https://example.com/f</code>
	1470	+	<code>%s --max-rate 1MB/s https://example.com/la</code>
1443	1471		<code>%s --plugin oauth2 https://api.example.com</code>
1444	1472		<code>%s --storage s3://mybucket/downloads/ http</code>
1445	1473		
-1449,5 +1477,5		<a href="#">View full file</a>	
1449	1477		<code>%s plugin enable oauth2</code>
1450	1478		<code>%s plugin config oauth2 --set client_id=xx</code>
1451	1479		

1452		- ` , appName, appName, appName, version, appNa
	1480	+ ` , appName, appName, appName, appName, vers
1453	1481	}

▼ internal/concurrent/worker.go

3 84.62% 55.56% -3.00%

-6,6 +6,8 <a href="#">View full file</a>		
6	6	"io"
7	7	"net/http"
8	8	"time"
	9	+
	10	+
9	11	)
10	12	
11	13	<b>type</b> Progress <b>struct</b> {
-17,12 +19,13 <a href="#">View full file</a>		
		}
17	19	
18	20	
19	21	<b>type</b> Worker <b>struct</b> {
20		- ID int
21		- Client *http.Client
22		- ChunkInfo *ChunkInfo
23		- URL string
24		- Progress chan<- Progress
25		- Error chan<- error
	22	+ ID int
	23	+ Client *http.Client
	24	+ ChunkInfo *ChunkInfo
	25	+ URL string
	26	+ Progress chan<- Progress
	27	+ Error chan<- error
	28	+ RateLimiter ratelimit.Limiter // Share
26	29	}
27	30	
28	31	// NewWorker creates a new download worker.
-49,7 +52,7 <a href="#">View full file</a>		
49	52	}
50	53	
51	54	// Try download with retry logic
52		- err := w.downloadChunk()
	55	+ err := w.downloadChunk(ctx)
53	56	<b>if</b> err != nil {
54	57	<b>if</b> w.Error != nil {
55	58	w.Error <- fmt.Errorf
-76,7 +79,7 <a href="#">View full file</a>		
76	79	}
77	80	
78	81	// downloadChunk performs the actual chunk c
79		- <b>func</b> (w *Worker) downloadChunk() error {
	82	+ <b>func</b> (w *Worker) downloadChunk(ctx context.C
80	83	maxRetries := 3
81	84	baseDelav := 100 * time.Millisecond



82	85	
-93,7 +96,7		<a href="#">View full file</a>
93	96	}
94	97	
95	98	// Attempt download
96		- err := w.performDownload()
	99	+ err := w.performDownload(ctx)
97	100	if err == nil {
98	101	return nil
99	102	}
-114,7 +117,7		<a href="#">View full file</a>
114	117	}
115	118	
116	119	// performDownload performs a single downloa
117		- func (w *Worker) performDownload() error {
	120	+ func (w *Worker) performDownload(ctx context
118	121	// Create range request
119	122	req, err := http.NewRequest("GET", w.l
! 120	! 123	if err != nil {
-145,6 +148,13		<a href="#">View full file</a>
145	148	// Read from response body
146	149	n, err := resp.Body.Read(buffe
147	150	if n > 0 {
	151	+ // Apply rate limiting
	! 152	+ if w.RateLimiter != n
! 153	+ 153	+ if rateLimiter
! 154	+ 154	+ return
! 155	+ 155	+ }
	156	+ }
	157	+ }
149	159	w.ChunkInfo.Downloaded
150	160	
151	161	// Send progress updat

> godl.go	1	60.14%	66.67%	+3.54%
> pkg/ratelimit/limiter.go	0	100.00%	100.00%	-
> pkg/types/types.go	0	100.00%	-	0.00%

