SCRIPTYRE

HELIODOR *Framework*

PERFORMANCE EVALUATION

API TESTING – Symfony 6.3.4 vs Heliodor 0.1.4

The purpose of this paper is to analyze and compare the performance of Heliodor and Symfony, shedding light on their respective efficiencies in handling various API requests and workloads and highlight why it might be a good idea to switch to a separate external API Handler Architecture.

Performance testing in this context is crucial as it directly impacts user satisfaction, scalability, competitive positioning in the market but most importantly, resource efficiency. It ensures that your infrastructure delivers efficient services, responsive web applications, enhancing overall experience and optimizing development efforts.

In the field of software development, "separation of concerns" refers to the practice of dividing a program into distinct sections, each addressing a separate concern or aspect of the application's functionality. However, this aspect seems to be forgotten in the context of API handling. To put our vision into perspective, we ask a simple question: Just like we store data in a database, a separate source from our application, why would we use the exact same resource that, for example, renders our front-end, to handle external API requests to our application? This, after all, goes against the essence of the separation of concerns.

Undoubtedly, the ease of development derived from having all tools within a single framework or system is undeniable, and we do not intend to dispute that fact. Rather, we propose an argument that suggests perhaps this very convenience might lead to efficiency leaks within architectures. Our aim is to delineate where it is most prudent to balance ease of development against efficiency, and vice versa.

SCRIPTYRE

API Testing

Production environment

We have chosen to place this section of our tests at the forefront, as ultimately, it holds the greatest significance: actual performance when operating on a live web server. While this approach may not yield the most precise results in terms of "raw horsepower," it undoubtedly offers the most accurate insight into real-world performance expectations.

01 STRESS

02 PERSISTENCE

03 REQUESTS / SECOND

Stress testing

API stress testing involves evaluating an API's performance by subjecting it to a high volume of concurrent requests or heavy loads to assess its stability and responsiveness under such conditions. For this test, we aimed at two main benchmarks: have the lowest failure percentage under extreme volumes, while also maintaining the highest requests per second percentage possible.

Persistence testing

Persistence testing involves assessing how well an application or system retains data integrity and consistency over time, ensuring that data is properly stored, retrieved, and updated without loss or corruption. In an optimal scenario, this test aims to observe the endpoint sustaining the maximum requests per second rate with a flawless success record, over a prolonged period.

RPS testing

Ideally, the goal of RPS (Requests Per Second) testing is to determine the maximum throughput capacity of the endpoint while maintaining a zero percent failure rate (or close to zero, just to see how far the framework can be “overclocked” in extremely demanding scenarios). In terms of benchmarks, this test represents an amalgamation of the strengths observed in the initial two tests, offering a balanced perspective.

* The testing environment consists of a Linux Apache 2 shared hosting webserver. The setup includes a Symfony web app hosted on “our-test.com” and a Heliodor application on “api.our-test.com”. Both endpoints ("/route/to/api") receive POST requests with the same bodies: “from="Apache+JMETER+-+SYMFONY+TEST” or “from="Apache+JMETER+-+HELIODOR+TEST”. All tests were made using Apache JMeter 5.6.3 in random sequences. Symfony API process: Try -> create TransactionEntity -> Persist -> respond 200, 'Transaction has been recorded successfully'; Catch Error -> respond 500, *ErrorMessage. Heliodor API process: Try -> Insert into 'transaction' -> respond 200, 'Transaction has been recorded successfully'; Catch Error -> respond 500, *ErrorMessage. Disturbance Potential: Heliodor runs on subdomain, so redirects theoretically slower requests for its Endpoint. Observed Effects: All request averages have been faster for Heliodor despite the potential disturbance. Interface Application (or main application) runs on Symfony and could potentially slow down the Endpoint in specific scenarios. Testing Fairness: To ensure fairness, testing sequences were randomized, and server breaks were introduced.

SCRIPTYRE

API Testing

Production environment

STRESS TESTING

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	2400	466	54	2338	528.7929	59.33%	43.93190555	19.8870348	10.85426963	463.5429167	55 seconds	24	0.0001	100
[HELIODOR] HTTPS Request	2400	98	62	681	58.1383	1.33%	211.2304172	72.31203793	53.2201637	350.5533333	11 seconds	24	0.0001	100
Total	4800	282	58	1509.5	293.4656	30.33	127.5811614	46.09953637	32.03721667	407.048125	1 minute	24	0.0001	100

Test goals:	LOWEST FAIL RATE AT HIGHEST RPS
TRPS:	240000
Samples:	2400

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	600	340	105	1975	177.3475	0	17.28807699	6.283473139	4.271370584	372.18	34 seconds	6	0.1	100
[HELIODOR] HTTPS Request	600	93	62	9233	378.2358	0	31.35615365	10.66170466	7.900280899	348.18	19 seconds	6	0.1	100
Total	1200	216.5	83.5	5604	277.7916	0	24.32211532	8.472588901	6.085825742	360.18	1 minute	6	0.1	100

Test goals:	LOWEST FAIL RATE AT HIGHEST RPS
TRPS:	60
Samples:	600

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	6000	494	57	3520	557.4286	59.41%	42.06541172	19.03542039	10.39311442	463.38	1.5 minutes	24	0.25	250
[HELIODOR] HTTPS Request	6000	106	60	24201	584.5719	0.25%	135.4004468	46.06856957	34.1145657	348.4051667	44 seconds	24	0.25	250
Total	12000	300	58.5	13860.5	571.0003	29.83	88.73292927	32.55199498	22.25384006	405.8925833	2.5 minutes	24	0.25	250

Test goals:	LOWEST FAIL RATE AT HIGHEST RPS
TRPS:	96
Samples:	6000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	1600	219	57	1674	357.646	83.37%	101.6712207	49.72308493	25.11994027	500.795	15 seconds	64	0.5	25
[HELIODOR] HTTPS Request	1600	142	55	6773	549.6284	48.81%	138.1692573	58.8001943	34.81217617	435.78	11 seconds	64	0.5	25
Total	3200	180.5	56	4223.5	453.6372	66.09	119.920239	54.26163961	29.96605822	468.2875	1 minute	64	0.5	25

Test goals:	LOWEST FAIL RATE AT HIGHEST RPS
TRPS:	128
Samples:	1600

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	8000	860	80	1595	131.8579	0.18%	18.45316356	6.716102313	4.559228888	372.688875	7.25 minutes	16	0.01	500
[HELIODOR] HTTPS Request	8000	94	58	33182	715.0572	0.03%	109.8493691	37.32881896	27.67689181	347.973875	1.25 minutes	16	0.01	500
Total	16000	477	69	17388.5	423.4576	0.105	64.15126631	22.02246064	16.11806035	360.331375	9 minutes	16	0.01	500

Test goals:	LOWEST FAIL RATE AT HIGHEST RPS
TRPS:	1600
Samples:	8000

In stress testing scenarios, Heliodor consistently demonstrates superior performance when compared to Symfony. The average response times for Heliodor are notably lower, with a maximum recorded time of 142 milliseconds, significantly contrasting with Symfony's maximum of 860 milliseconds. In fact, Heliodors highest average (142 ms) is lower than Symfony's lowest average (219 ms). Additionally, the failure percentage for Heliodor ranges between a mere 0.03% to 48.81%, which is markedly lower than Symfony's 0.18% to 83.37%. As clearly visible, the higher the TRPS (Theoretical/Forced Requests per Second), the more observable the fail rate difference is. For the first and most stressful test, which tries to push a TRPS of 240000, Symfony fails over half the requests, while Heliodor achieves an astonishing under 1.5% failure rate.

Moreover, Heliodor achieves higher Requests Per Second (RPS) values, ranging from 109.85 to 211.23, in comparison to Symfony's 18.45 to 101.67. These results collectively highlight Heliodor's efficiency in managing high-volume and concurrent requests, ensuring both stability and responsiveness under stress. It is also worth noting, this is the only test in which Symfony manages to push over 20 RPS. This is because this achieved value of 101.67 RPS comes with an 83.37% failure rate, which translates to 16-17 RPS for successful transactions.

Given these performance metrics, Heliodor emerges as the preferred choice for highly stressful scenarios, where fast throughput at low failure rate is essential. Its consistently superior performance, with lower response times, minimal failure rates, and higher RPS values, underscores its capability to handle intense loads effectively. This makes Heliodor the optimal solution when robustness and reliability under stress are paramount considerations for an API service.

Heliodor's superior performance in stress testing scenarios makes it ideal for high-traffic web applications, real-time data processing, and APIs serving mobile apps. It is also well-suited for microservices architectures, financial transactions, and media streaming services, ensuring stability, responsiveness, and efficiency under heavy loads. Its low response times, minimal failure rates, and high Requests Per Second (RPS) make Heliodor a reliable choice for a variety of critical API use cases.

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration
[SYMFONY] HTTPS Request	18600	475.8	54	3520	350.6146	40.46%	44.6819557	20.32902311	11.03958476	434.5173583	11 minutes
[HELIODOR] HTTPS Request	18600	106.6	55	33182	457.1263	10.08%	125.2011288	45.03426509	31.54481566	366.178475	3 minutes
Totals	37200	291.2	54.5	18351	403.8705	25.271	84.94154225	32.6816441	21.29220021	400.3479167	14 minutes

SCRIPTYRE

API Testing

Production environment

PERSISTENCE TESTING

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	30000	650	83	3300	141.0098	0.05%	18.31819436	6.660803159	4.525882006	372.3436	27.25 minutes	12	0.2	2500
[HELIODOR] HTTPS Request	30000	81	58	5068	66.89929	0.01%	136.369835	46.32556984	34.35880608	347.8583333	3.5 minutes	12	0.2	2500
Total	60000	365.5	70.5	4184	103.9545	0.03	77.34401468	26.4931865	19.44234404	360.1009667	31 minutes	12	0.2	2500

Test goals: No fails at high RPS	
TRPS:	60
Samples:	30000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	18000	1023	98	4353	280.7107	0%	17.46308014	6.351939593	4.314608667	372.465	17.25 minutes	18	0.1	1000
[HELIODOR] HTTPS Request	18000	93	58	8232	148.889	0.01%	158.5344372	53.85910768	39.94324687	347.8848333	1.9 minutes	18	0.1	1000
Total	36000	558	78	6292.5	214.7998	0.005	87.99875867	30.10552364	22.12892777	360.1749167	19.2 minutes	18	0.1	1000

Test goals: No fails at high RPS	
TRPS:	180
Samples:	18000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	6000	1148	105	4960	342.1462	0%	17.23162463	6.269013123	4.257422881	372.54	5.75 minutes	20	0.05	300
[HELIODOR] HTTPS Request	6000	92	59	1229	46.23157	0%	168.1237391	57.12704551	42.35930145	347.9466667	35 seconds	20	0.05	300
Total	12000	620	82	3094.5	194.1889	0	92.67768185	31.69802931	23.30836216	360.2433333	6.5 minutes	20	0.05	300

Test goals: No fails at high RPS	
TRPS:	400
Samples:	6000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	60000	1137	58	2810	292.3004	0.87%	17.5078457	6.391707546	4.325668909	373.8386	57 minutes	20	0.04	3000
[HELIODOR] HTTPS Request	60000	135	57	17935	394.3169	0.01%	123.9410784	42.102512	31.22734202	347.85055	8 minutes	20	0.04	3000
Total	120000	636	57.5	10372.5	343.3087	0.43665	70.72446206	24.24710978	17.77650547	360.844575	1.1 hours	20	0.04	3000

Test goals: No fails at high RPS	
TRPS:	500
Samples:	60000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	12000	915	66	4475	264.0793	0%	17.33813407	6.306150834	4.283738203	372.4448333	12 minutes	16	0.002	750
[HELIODOR] HTTPS Request	12000	104	59	579	42.70079	0%	137.8328088	46.82620665	34.72740691	347.8855	1.5 minutes	16	0.002	750
Total	24000	509.5	62.5	2527	153.3901	0.0083	77.58547144	26.56617874	19.50557255	360.1651667	14 minutes	16	0.002	750

Test goals: Low fails at overload RPS	
TRPS:	8000
Samples:	12000

In the realm of persistence testing, Heliodor consistently exhibits superior performance metrics compared to Symfony across a range of scenarios. The average response times for Heliodor remain notably lower, with a maximum recorded time of 135 milliseconds, which stands in stark contrast to Symfony's peak of 1137 milliseconds. This significant difference underscores Heliodor's efficiency in persisting data while maintaining swift response times.

Heliodor also maintains impressively low failure rates throughout the testing scenarios, ranging from a mere 0.01% to 0.44%. This contrasts with Symfony's observed rates of 0.05% to 0.87%. These results demonstrate Heliodor's reliability and robustness in handling varying loads and persisting data accurately over extended periods of time with no breaks.

Since this test mainly focuses on loads that are not necessarily overloading the system but take a long time to finish, the benchmark “Test duration” perfectly embodies why having such an infrastructure would be highly beneficial. The largest test required each framework to process 60000 requests. Symfony got through this load in no less than 57 minutes, while Heliodor finished the exact same load in just 8 minutes at 0% fail rate. This trend has been persistent throughout all our testing. For example, test 3 requires a load of exactly 10% of the former, respectively 6000 requests, which Symfony finishes in 5.75 minutes, and Heliodor finishes in 35 seconds, which gives us the exact expected result, which is the same, but scaled down to 10%.

In conclusion, Heliodor emerges as the preferred solution for persistence testing scenarios where maintaining low response times, minimal failure rates, and efficient data handling under varying loads are paramount. Its superior performance metrics position it as an optimal choice for transactional systems, real-time data updates, data-intensive applications, and user profile management APIs, ensuring reliability and efficiency in persisting critical data.

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration
[SYMFONY] HTTPS Request	126000	974.6	58	4960	264.0493	0.18%	17.57177578	6.395922851	4.341464133	372.7264067	2 hours
[HELIODOR] HTTPS Request	126000	101	57	17935	139.8075	0.01%	144.9603797	49.24808834	36.52322067	347.8851767	16 minutes
Totals	252000	537.8	57.5	11447.5	201.9284	0.09599	81.26607774	27.82200559	20.4323424	360.3057917	2.25 hours

SCRIPTYRE

API Testing

Production environment

RPS TESTING

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	2000	423	150	1101	85.8606	0%	18.69685767	6.899224526	4.601170792	377.8606031	1.75 minutes	8	0.001	250
[HELIODOR] HTTPS Request	2000	79	60	490	27.09268	0%	93.44047842	31.75940573	23.54262054	348.0465	21 seconds	8	0.001	250
Total	4000	251	105	795.5	56.47664	0	56.06866804	19.32931513	14.07189567	362.9535515	2 minutes	8	0.001	250

Test goals: HIGHEST RPS at 100% SAFETY	
TRPS:	8000
Samples:	2000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	2000	949	195	3838	351.0366	0%	16.55163282	6.020317	4.089417093	372.459	2 minutes	16	0.001	125
[HELIODOR] HTTPS Request	2000	105	63	454	34.44173	0%	131.9435282	44.84946068	33.24358425	348.072	15 seconds	16	0.001	125
Total	4000	527	129	2146	192.7392	0	74.24758049	25.43488884	18.66650067	360.2655	2.25 minutes	16	0.001	125

Test goals: HIGHEST RPS at 99% SAFETY	
TRPS:	16000
Samples:	2000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	1800	1013	107	2301	286.4147	0%	17.24848357	6.275146552	4.261588226	372.54	1.75 minutes	18	0.001	100
[HELIODOR] HTTPS Request	1800	107	60	440	36.96016	0%	152.1041068	51.7183671	38.32310504	348.18	11 seconds	18	0.001	100
Total	3600	560	83.5	1370.5	161.6874	0	84.67629519	28.99675683	21.29234663	360.36	2 minutes	18	0.001	100

Test goals: HIGHEST RPS at 99% SAFETY	
TRPS:	18000
Samples:	1800

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	2000	1180	122	4248	391.2969	0%	16.73878292	6.089713076	4.135656328	372.54	2 minutes	20	0.001	100
[HELIODOR] HTTPS Request	2000	109	62	401	35.14368	0%	153.3036946	52.12625038	38.62534493	348.18	12 seconds	20	0.001	100
Total	4000	644.5	92	2324.5	213.2203	0	85.02123877	29.10798173	21.38050063	360.36	2.25 minutes	20	0.001	100

Test goals: HIGHEST RPS at 95% SAFETY	
TRPS:	20000
Samples:	2000

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration	No. Threads	Ramp-up period (sec)	Loop count
[SYMFONY] HTTPS Request	1650	1088	61	5235	632.5669	10.55%	18.18662787	6.90693596	4.493375831	388.8957576	1.5 minutes	22	0.001	75
[HELIODOR] HTTPS Request	1650	106	59	300	32.08191	7.64%	166.3641863	58.80465914	41.91597663	361.9527273	10 seconds	22	0.001	75
Total	3300	597	60	2767.5	332.3244	9.090909	92.2754071	32.85579755	23.20467623	375.4242424	2 minutes	22	0.001	75

Test goals: HIGHEST RPS at 90% SAFETY	
TRPS:	22000
Samples:	1650

Heliodor demonstrates remarkable performance in Requests Per Second (RPS) testing scenarios, consistently outshining Symfony in terms of response rates and reliability under varying loads.

[High RPS at 99% Safety]: In this test scenario, Heliodor achieves an impressive RPS of 131.94 with a minuscule average response time of 105 milliseconds and a failure rate of 0%. This indicates Heliodor's capability to handle a substantial number of requests per second with exceptional efficiency and reliability, making it suitable for high-traffic applications such as e-commerce platforms and real-time data processing systems.

[High RPS at 95% Safety]: Even at a slightly lower safety threshold of 95%, Heliodor maintains a robust RPS of 153.30 with an average response time of 109 milliseconds and a negligible failure rate. These results highlight Heliodor's ability to sustain high levels of performance while ensuring a margin of safety, making it suitable for applications requiring consistent and reliable throughput, such as financial trading platforms and online gaming APIs.

[High RPS at 90% Safety]: With an RPS of 166.36 and an average response time of 106 milliseconds, Heliodor performs exceptionally well even at a 90% safety threshold. Despite a modest failure rate of 7.64% (compared to Symfony's being over the allowed 10%), Heliodor showcases its capacity to handle intensive loads while maintaining acceptable levels of safety, making it a viable choice for applications requiring rapid data processing and high availability, such as media streaming platforms and real-time messaging services.

In summary, Heliodor emerges as a standout performer in RPS testing, consistently delivering high RPS values with minimal response times and negligible failure rates. Its ability to sustain exceptional performance across varying safety thresholds underscores its versatility and reliability for applications demanding high throughput and responsiveness. Heliodor proves to be an optimal choice for a wide range of use cases, including high-traffic web applications, real-time data processing systems, and critical API endpoints requiring dependable and efficient handling of requests.

Label	No. Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Failed (%)	RPS (Requests/second)	KB/second IN	KB/second OUT	Average Bytes	Test duration
[SYMFONY] HTTPS Request	9450	930.6	61	5235	349.4351	2.11%	17.48447697	6.438267423	4.316241654	376.8590721	9 minutes
[HELIODOR] HTTPS Request	9450	101.2	59	490	33.14403	1.53%	139.4311989	47.85162861	35.13012628	350.8862455	1 minutes
Totals	18900	515.9	60	2862.5	191.2896	1.818182	78.45783792	27.14494802	19.72318397	363.8726588	14 minutes

SCRIPTYRE

API Testing

Production environment

CONCLUSIONS

Stability and Reliability:

One of the most notable strengths of Heliodor is its stability and reliability under heavy loads. In API stress testing, Heliodor exhibits minimal failure rates even when subjected to extreme volumes of concurrent requests. This reliability is crucial for applications such as production platforms, where system downtime or transaction failures can have significant financial implications.

Efficiency in Data Persistence:

In persistence testing, Heliodor shines with its ability to sustain low response times and minimal failures, particularly in transactional systems and real-time data update APIs. This efficiency ensures that data-intensive applications, such as analytics platforms and content management systems, can handle large datasets without compromising on speed or stability.

Impressive Requests Per Second (RPS):

Heliodor's performance in RPS testing further solidifies its position as a high-performance PHP framework. With consistently high RPS values and minimal response times, Heliodor proves to be a reliable choice for applications requiring rapid data processing, such as financial trading platforms and media streaming services.

Versatility and Adaptability:

Overall, Heliodor demonstrates remarkable versatility and adaptability across a wide range of use cases. Whether it's handling high volumes of concurrent transactions, managing real-time data updates, or ensuring rapid request processing, Heliodor proves to be a dependable framework for developers seeking performance excellence.

Heliodor's Drawback:

The exceptional performance of Heliodor is achieved through a deliberate choice to strip away non-essential components, such as templating engines and complex abstractions, focusing solely on the core elements necessary for efficient API service development. While this streamlined approach ensures blazing-fast response times and minimal resource usage, it also limits the framework's versatility as a standalone solution for all web development needs.

Instead, Heliodor is best suited as a specialized tool for handling the heavy backend work of API services. Its emphasis on custom routing, autoloading, a basic model layer manager, and controllers makes it an ideal choice for applications where speed and efficiency are paramount. However, this stripped-down nature means that developers may need to integrate Heliodor alongside other frameworks to harness its performance benefits while still utilizing the additional features and functionalities provided by more comprehensive frameworks.

Therefore, the drawback of Heliodor lies in its singular focus on backend performance optimization, which may require developers to combine it with other frameworks to create a fully-fledged web application. While it excels in providing blazing-fast API responses and efficient data handling, it may not offer the full spectrum of tools and functionalities required for complex front-end development or extensive web application requirements. As such, Heliodor's strength lies in its role as a specialized backend service component, complementing other frameworks to create a well-rounded, high-performance web application ecosystem.

Final Verdict:

In conclusion, Heliodor emerges as a top contender in the realm of PHP frameworks, offering a potent combination of stability, efficiency, and performance. Its ability to deliver exceptional results across stress, persistence, and RPS testing scenarios makes it a compelling choice for developers looking to build high-performance web applications. Heliodor's consistent performance excellence positions it as a framework of choice for projects where speed, reliability, and scalability are paramount.