

Online Appendices for: Do Fewer Tiers Mean Fewer Tears?

This document contains the online appendices for "Fewer Tiers Mean Fewer Tears? Eliminating Web Stack Components to Improve Interoperability". The appendices cover code examples, comparison tables, and additional performance measurements.

A Developer Effort - Links vs PHP

Although tierless languages generally simplify the development of web applications, there are some cases that are less than elegant, as discussed in Section 2.2 of [1]. An example is database update operations in Links where all references to a Table Handle must include all the fields in the table. This introduces a large amount of redundancy in the code as it is not often that all columns in a database are updated simultaneously.

Listing 1. PHP Code Example

```
function markMilestoneComplete ($id) { mysql query("UPDATE milestone SET completed
    = 1 WHERE milestoneid = $id")};
```

Listing 2. Links Code Example

```
fun markMilestoneComplete (milestoneid) {
update (var milestone <-- tb milestone)
where (milestone.milestoneid == milestoneid)
set (
milestoneid = milestone.milestoneid,
projectid = milestone.projectid,
creator = milestone.creator,
summary = milestone.summary,
priority = milestone.priority,
date start = milestone.date start,
date end = milestone.date end,
completed = 1
);
}
```

The one line SQL statement using PHP is easy to comprehend and makes it clear which fields are being updated, whereas the Links code requires 14 lines to maintain clarity, yet the columns being updated are far less obvious. Even if only updating a single column, Links requires code to be written relating to all columns in the table definition, increasing developer effort.

B Apache & Jupyter Interoperation

In JAPyL, (Section 2.3 [1]), the Jupyter Notebook is embedded in the webpage via HTML using the IFRAME Tag. However, by default for security reasons, the Jupyter Notebook does not allow its Notebooks to be embedded externally.

This feature has to be enabled in the Jupyter Server using Jupyter Configs in the `jupyter_notebook_config.py` configuration file.

Listing 3. Jupyter Configuration for Enabling Notebook Embedding

```
c.NotebookApp.ip = '0.0.0.0'
c.NotebookApp.tornado_settings = {
'headers':{
```

```

    'Content-Security-Policy': "frame-ancestors https://192.168.43.100:10125 'self' "
  }
}
c.NotebookApp.base_url = '/jupyter/notebooks/'
c.NotebookApp.open_browser = False
c.NotebookApp.port = 9253

```

The Apache web server is then configured to act as a reverse proxy. The configuration is implemented in `/etc/apache2/sites-available/default-ssl.conf`.

Listing 4. Reverse Proxy Configuration in Apache

```

<IfModule mod_ssl.c>
    <VirtualHost default_:10125>

ServerAdmin email@example.net
ServerName 192.168.43.125

LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule ssl_module modules/mod_ssl.so

DocumentRoot /var/www/html

<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

SSLEngine on
SSLProxyEngine On
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off
SSLCertificateFile /etc/apache2/ssl/certificates/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/certificates/apache.key

<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

ProxyPass / https://192.168.43.125:10125/
ProxyPassReverse / https://192.168.43.125:10125/

```

```
</VirtualHost>
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

C JAPyL Declarative Security Examples

In JAPyL, the security mechanisms are typically declaratively specified in either the Jupyter Server or Apache as discussed in Section 3.1.2 [1]. Two examples demonstrated below include IP Whitelisting, Blacklisting and Security Headers implementation.

Listing 5. Apache IP Whitelisting, Blacklisting

```
<Directory>

    Order Allow,Deny
    Allow from 192.168.43.15

</Directory>
```

Listing 6. Apache Security Headers

```
<IfModule mod_headers.c>

strict-transport-security: max-age=31536000; includeSubDomains; preload
expect-ct: max-age=31536000; enforce
referrer-policy: no-referrer
referrer-policy: strict-origin-when-cross-origin
x-permitted-cross-domain-policies: none
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
x-frame-options: allow

</IfModule>
```

D Developer Effort Schematic

Tables 1 & 2 provide a summary of the files and languages a developer needs to edit or implement in both JAPyL & JPL web stacks.

Table 1. JAPyL Implementation

Component/Language	Functionality	Configuration File	Implementation
Apache	Host Web Page to Embed Notebook	apache2.conf	Apache Configs
	Reverse Proxy Settings	default-ssl.conf	Apache Configs
	Apache Security Settings	.htaccess file	Apache Configs
Python (Kernel)	Language Processing	ipykernel.py	Python
	Messaging & Communication	kernel.json	JSON
Webpage	Embed Notebook	index.html	HTML

Table 2. JPL Implementation

Component/Language	Functionality	Configuration File	Implementation
PHP	Host Web Page to Embed Notebook	react.php	PHP
	Reverse Proxy Settings	proxy.php	PHP
	Apache Security Settings	security.php file	PHP
	Language Processing	kernelcore.php	PHP
	Messaging & Communication	kerneloutput.php	PHP
Webpage	Embed Notebook	index.php	PHP

E JPL Imperative Security Examples

In JPL, the security mechanisms are typically imperatively specified as discussed in Section 3.2.1 [1]. Two examples demonstrated below include IP Whitelisting, Blacklisting and Security Headers implementation.

Listing 7. PHP IP Whitelisting, Blacklisting

```
<?php

$deny = array("192.168.43.20", "192.168.43.60", "192.168.43.75");
if (in_array($_SERVER['REMOTE_ADDR'], $deny)) {
    header("location: https://192.168.43.125:10125/");
    exit();
}

?>

<?php

function isAllowed($ip){
    $whitelist = array('192.168.43.70', '192.168.43.130', '192.168.43.54');
```

```
// If the ip is matched, return true
if(in_array($ip, $whitelist)) {
    return true;
}

foreach($whitelist as $i){
    $wildcardPos = strpos($i, "*");

    // Check if the ip has a wildcard
    if($wildcardPos !== false && substr($ip, 0, $wildcardPos) . "*" == $i) {
        return true;
    }
}

return false;
}

?>
```

Listing 8. PHP Security Headers

```
require_once('SecureHeaders.phar');

use Aidantwoods\SecureHeaders\SecureHeaders;

$headers = new SecureHeaders();
$headers->hsrs();
$headers->csp('default', 'self');
$headers->csp('script', 'https://192.168.43.140:10125');
$headers->apply();
```

F Performance Results: Other Platforms

Performance results in terms of latency & throughput (Figures 1, 2, 3, 4) were almost similar to those on the Docker platform, as discussed in Section 4.2 [1], where JAPyL outperformed JPL.

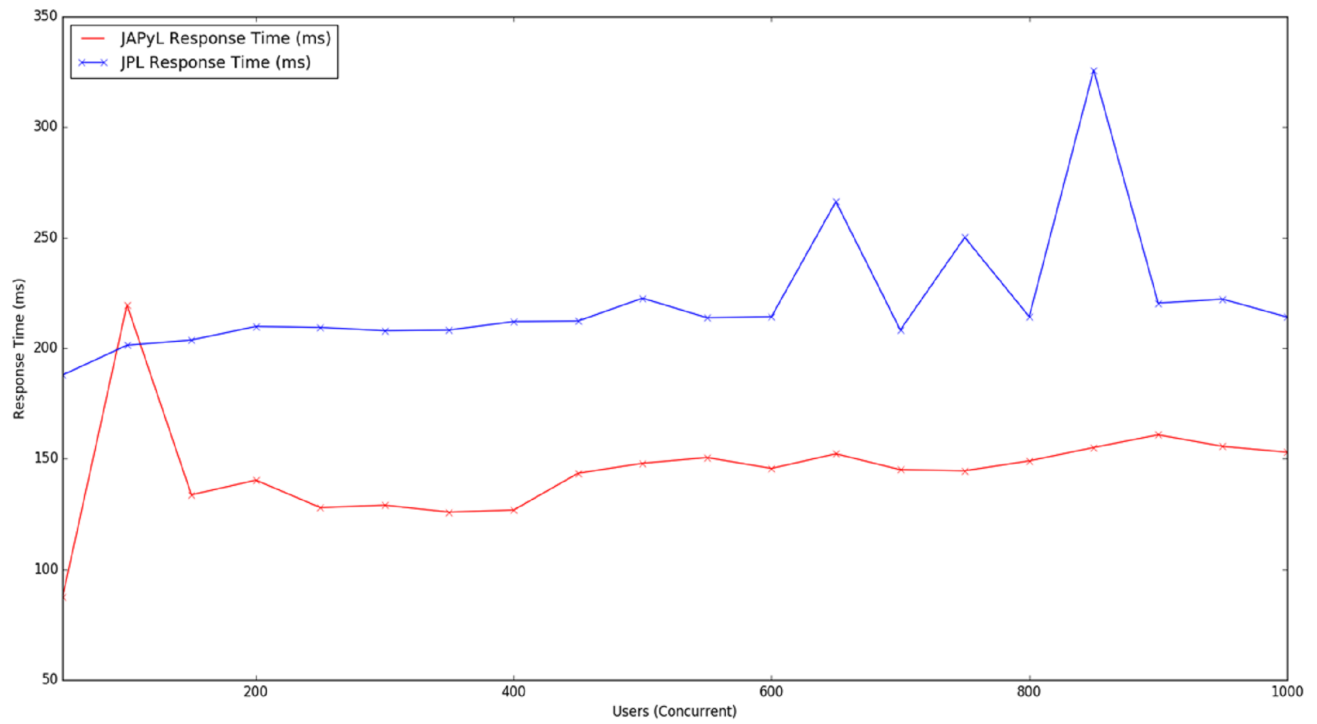


Figure 1. JAPyL vs JPL Latencies (VirtualBox)

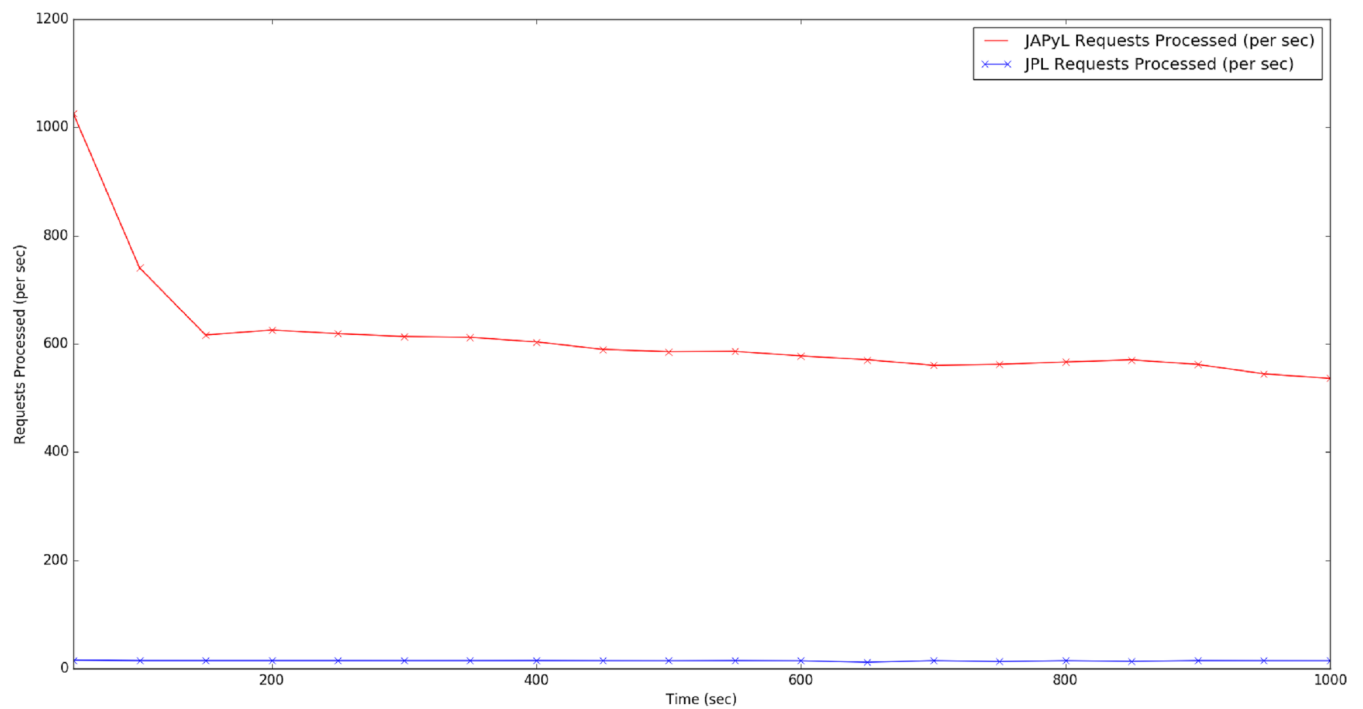


Figure 2. JAPyL vs JPL Throughput (VirtualBox)

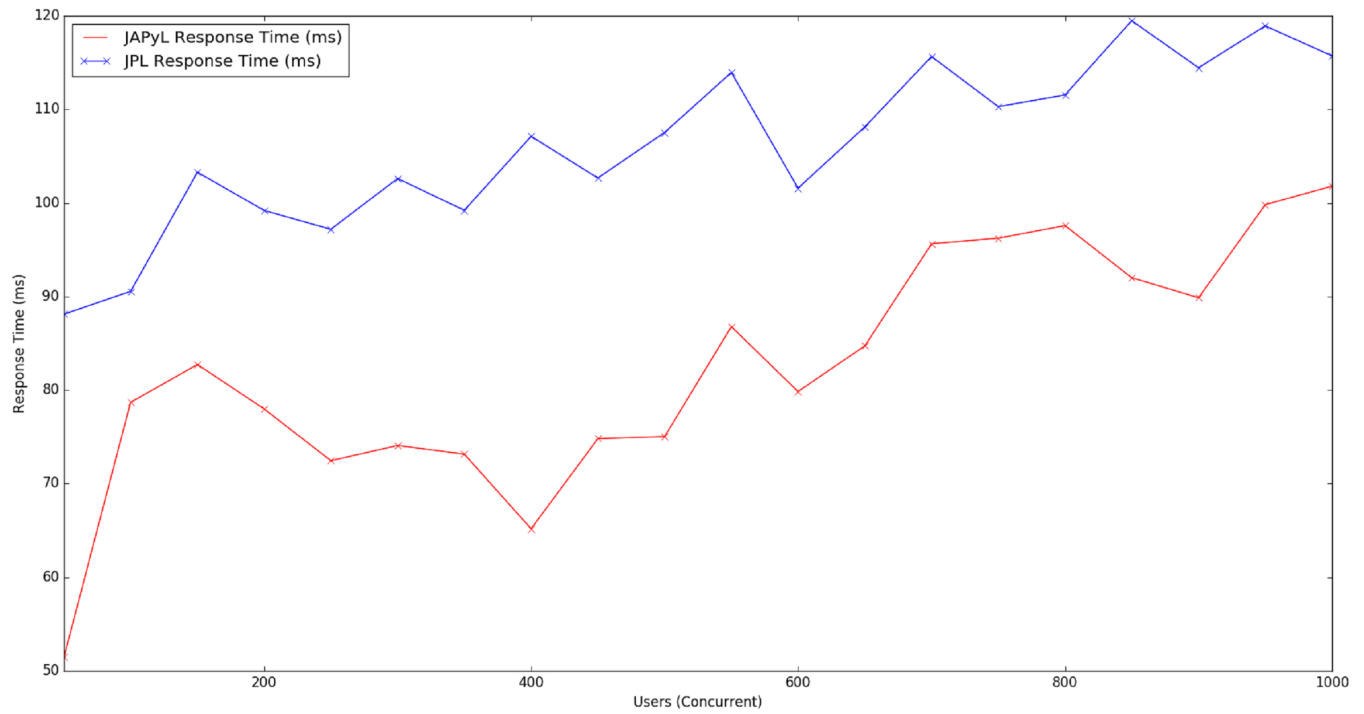


Figure 3. JAPyL vs JPL Latencies (Raspberry Pi)

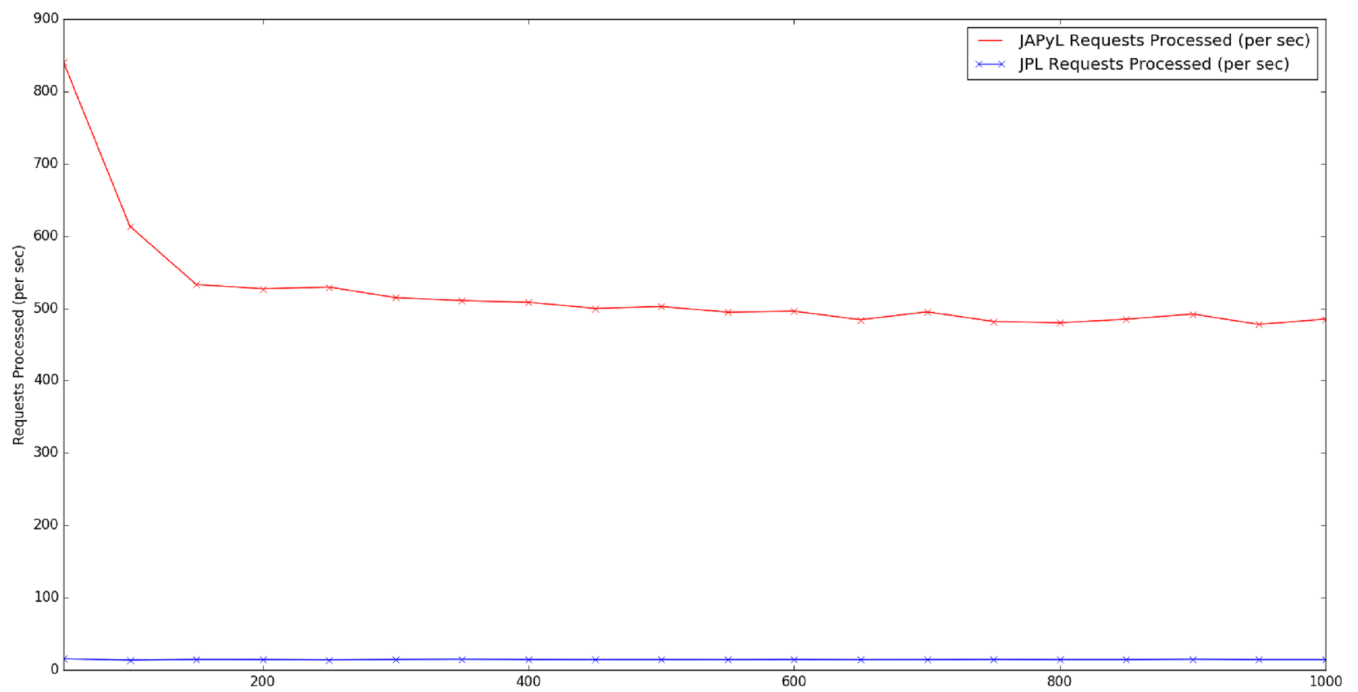


Figure 4. JAPyL vs JPL Throughput (Raspberry Pi)

References

- [1] Ramsingh Adrian, Singer Jeremy, and Phil Trinder. 2020. Online Appendix: Do Fewer Tiers Mean Fewer Tears? Retrieved Jan 14, 2020 from <http://www.dcs.gla.ac.uk/~ramsad/papers/FewerTiers.pdf>