benchmark_k6_write

July 13, 2025

[8]:
```
import json
import glob
import re
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Suffix for selecting benchmark results
suffix = "/write_avgnet/"

# Path to your benchmark results directory
results_dir = './results_store/_final/'
ec_pattern = results_dir + f'erasure{suffix}/_write_*_*.json'
replication_pattern = results_dir + f'replication{suffix}/_write_*_*.json'

# Helper to extract data from files
def extract_data(files, system_type):
    data = []
    for file in files:
        # Updated regex: matches kbit, mbit, gbit
        match = re.search(r'_write_(.+?)_(\d+)vu(?:_([0-9]+(?:kbit|mbit|gbit)))?
 ↪\.json', file)
        if not match:
            continue
        payload_size = match.group(1)
        if payload_size.endswith('b'):
            payload_size = payload_size[:-1]
        try:
            payload_size = int(payload_size)
        except ValueError:
            pass
        virtual_user = match.group(2)
        bandwidth = match.group(3) if match.group(3) else 'unlimited'
        with open(file) as f:
            j = json.load(f)
            # try summary.success_performance, else fallback to details.
 ↪http_req_duration
```

```
            sp = j.get('summary', {}).get('success_performance')
            if sp:
                med = sp.get('med', 0)
                p90 = sp.get('p(90)', 0)
                avg = sp.get('avg', 0)
            else:
                dur = j.get('details', {}).get('http_req_duration', {}).
 ↪get('values', {})
                med = dur.get('med', 0)
                p90 = dur.get('p(90)', 0)
                avg = dur.get('avg', 0)
            # extract request rate
            rate = j.get('summary', {}).get('reqs', {}).get('rate', 0)
            data.append({
                'system': system_type,
                'payload_size': payload_size,
                'virtual_user': int(virtual_user),
                'bandwidth': bandwidth,
                'med': med,
                'p90': p90,
                'avg': avg,
                'rate': rate
            })
    return data

def bandwidth_to_num(bw):
    if bw == 'unlimited':
        return float('inf')
    m = re.match(r'(\d+)(kbit|mbit|gbit)', bw)
    if not m:
        return float('inf')
    val, unit = int(m.group(1)), m.group(2)
    if unit == 'kbit':
        return val
    elif unit == 'mbit':
        return val * 1000
    elif unit == 'gbit':
        return val * 1000 * 1000
    return float('inf')

# Collect EC and Replication data
files_ec = glob.glob(ec_pattern)
files_replication = glob.glob(replication_pattern)
data = extract_data(files_ec, 'EC') + extract_data(files_replication,␣
 ↪'Replication')
df = pd.DataFrame(data)
if not df.empty:
```

```
    # Add bandwidth_num for correct sorting
    df['bandwidth_num'] = df['bandwidth'].apply(bandwidth_to_num)
    # Sort by payload_size, virtual_user, bandwidth_num, then system for␣
 ↪grouped comparison
    df = df.sort_values(['payload_size', 'virtual_user', 'bandwidth_num',␣
 ↪'system'])
    df['combo'] = df.apply(lambda row:␣
 ↪f"{row['payload_size']}B_{row['virtual_user']}vu_{row['bandwidth']}", axis=1)
    df.reset_index(drop=True, inplace=True)
df
```

[8]:

	system	payload_size	virtual_user	bandwidth	med	\
0	EC	200000	1	10mbit	699.992802	
1	Replication	200000	1	10mbit	1073.896833	
2	EC	200000	1	25mbit	341.796174	
3	Replication	200000	1	25mbit	429.650143	
4	EC	200000	1	40mbit	261.041856	
5	Replication	200000	1	40mbit	268.192058	
6	EC	200000	1	55mbit	236.579951	
7	Replication	200000	1	55mbit	194.562242	
8	EC	200000	1	70mbit	224.866249	
9	Replication	200000	1	70mbit	152.595633	
10	EC	400000	1	10mbit	1346.515693	
11	Replication	400000	1	10mbit	2141.258066	
12	EC	400000	1	25mbit	611.797533	
13	Replication	400000	1	25mbit	856.185386	
14	EC	400000	1	40mbit	441.182746	
15	Replication	400000	1	40mbit	526.053664	
16	EC	400000	1	55mbit	370.781581	
17	Replication	400000	1	55mbit	380.759582	
18	EC	400000	1	70mbit	335.465073	
19	Replication	400000	1	70mbit	295.433864	
20	EC	600000	1	10mbit	1994.064533	
21	Replication	600000	1	10mbit	3207.928697	
22	EC	600000	1	25mbit	874.594191	
23	Replication	600000	1	25mbit	1279.382928	
24	EC	600000	1	40mbit	615.983084	
25	Replication	600000	1	40mbit	800.167324	
26	EC	600000	1	55mbit	507.199511	
27	Replication	600000	1	55mbit	569.969124	
28	EC	600000	1	70mbit	458.925299	
29	Replication	600000	1	70mbit	443.692196	
30	EC	800000	1	10mbit	2640.590774	
31	Replication	800000	1	10mbit	4575.859446	
32	EC	800000	1	25mbit	1138.658848	
33	Replication	800000	1	25mbit	1682.151866	
34	EC	800000	1	40mbit	781.503670	

35	Replication	800000	1	40mbit	1035.841426
36	EC	800000	1	55mbit	628.045337
37	Replication	800000	1	55mbit	754.886365
38	EC	800000	1	70mbit	544.997969
39	Replication	800000	1	70mbit	581.446226
40	EC	1000000	1	10mbit	3281.701913
41	Replication	1000000	1	10mbit	4126.683997
42	EC	1000000	1	25mbit	1394.745454
43	Replication	1000000	1	25mbit	2100.551557
44	EC	1000000	1	40mbit	969.631511
45	Replication	1000000	1	40mbit	1298.559282
46	EC	1000000	1	55mbit	760.338166
47	Replication	1000000	1	55mbit	940.436655
48	EC	1000000	1	70mbit	656.293743
49	Replication	1000000	1	70mbit	722.669057

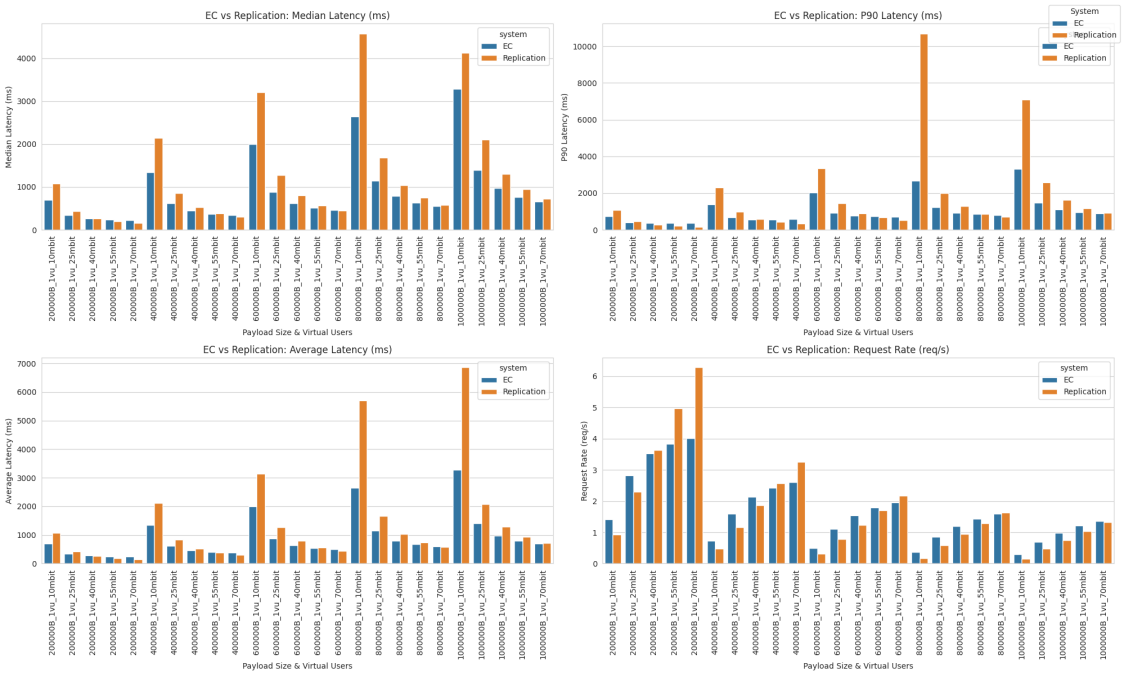
	p90	avg	rate	bandwidth_num	combo
0	723.726307	701.948700	1.409262	10000	200000B_1vu_10mbit
1	1086.421647	1065.650607	0.931688	10000	200000B_1vu_10mbit
2	399.206527	347.620375	2.816840	25000	200000B_1vu_25mbit
3	466.753251	427.348248	2.297152	25000	200000B_1vu_25mbit
4	360.153438	275.510498	3.531695	40000	200000B_1vu_40mbit
5	289.197103	266.721019	3.639365	40000	200000B_1vu_40mbit
6	353.503377	253.428548	3.829072	55000	200000B_1vu_55mbit
7	207.905265	193.301509	4.969445	55000	200000B_1vu_55mbit
8	365.015640	241.722793	4.010621	70000	200000B_1vu_70mbit
9	163.087706	151.345583	6.277621	70000	200000B_1vu_70mbit
10	1380.522840	1349.873754	0.733990	10000	400000B_1vu_10mbit
11	2299.629240	2110.231205	0.470518	10000	400000B_1vu_10mbit
12	672.821147	616.714089	1.590676	25000	400000B_1vu_25mbit
13	982.572614	846.049949	1.166899	25000	400000B_1vu_25mbit
14	547.281528	457.927517	2.138820	40000	400000B_1vu_40mbit
15	594.218360	526.925748	1.862124	40000	400000B_1vu_40mbit
16	566.006723	402.389760	2.422453	55000	400000B_1vu_55mbit
17	415.795460	381.054784	2.559696	55000	400000B_1vu_55mbit
18	572.220025	374.013042	2.604556	70000	400000B_1vu_70mbit
19	325.836070	296.484580	3.253635	70000	400000B_1vu_70mbit
20	2021.718340	1992.436063	0.496600	10000	600000B_1vu_10mbit
21	3333.020765	3140.549373	0.316151	10000	600000B_1vu_10mbit
22	933.686783	879.782181	1.116080	25000	600000B_1vu_25mbit
23	1431.338001	1265.027102	0.778043	25000	600000B_1vu_25mbit
24	771.424799	636.473482	1.532572	40000	600000B_1vu_40mbit
25	891.451074	790.580834	1.241447	40000	600000B_1vu_40mbit
26	735.781840	543.572890	1.790709	55000	600000B_1vu_55mbit
27	665.196284	570.356202	1.706402	55000	600000B_1vu_55mbit
28	720.457737	496.046357	1.953569	70000	600000B_1vu_70mbit
29	518.723513	445.896458	2.171622	70000	600000B_1vu_70mbit

30	2676.778082	2640.417792	0.374630	10000	800000B_1vu_10mbit
31	10689.518645	5703.423259	0.174301	10000	800000B_1vu_10mbit
32	1214.315463	1145.674224	0.852595	25000	800000B_1vu_25mbit
33	1983.821932	1671.141309	0.588509	25000	800000B_1vu_25mbit
34	930.897583	805.201959	1.195295	40000	800000B_1vu_40mbit
35	1282.618340	1035.320619	0.938890	40000	800000B_1vu_40mbit
36	845.940564	669.386850	1.433106	55000	800000B_1vu_55mbit
37	873.146131	747.611853	1.288427	55000	800000B_1vu_55mbit
38	811.969817	598.569975	1.594321	70000	800000B_1vu_70mbit
39	699.595252	579.789730	1.639239	70000	800000B_1vu_70mbit
40	3309.682499	3282.157200	0.301631	10000	1000000B_1vu_10mbit
41	7102.105745	6856.459491	0.145046	10000	1000000B_1vu_10mbit
42	1467.124810	1401.247301	0.696896	25000	1000000B_1vu_25mbit
43	2584.093762	2081.660721	0.471932	25000	1000000B_1vu_25mbit
44	1092.546838	975.910794	0.987347	40000	1000000B_1vu_40mbit
45	1617.774552	1287.491610	0.755246	40000	1000000B_1vu_40mbit
46	947.808956	787.650352	1.212604	55000	1000000B_1vu_55mbit
47	1161.139322	932.183974	1.032365	55000	1000000B_1vu_55mbit
48	903.999017	701.980075	1.354233	70000	1000000B_1vu_70mbit
49	914.024479	722.928048	1.316559	70000	1000000B_1vu_70mbit

[9]:
```
# Plotting: Compare EC vs Replication for each metric, grouped by␣
 ↪(payload_size, virtual_user)
sns.set_style("whitegrid")
metrics = ['med', 'p90', 'avg', 'rate']
metric_titles = {
    'med': 'Median Latency (ms)',
    'p90': 'P90 Latency (ms)',
    'avg': 'Average Latency (ms)',
    'rate': 'Request Rate (req/s)'
}

fig, axes = plt.subplots(2, 2, figsize=(20, 12))
axes = axes.flatten()

for ax, metric in zip(axes, metrics):
    sns.barplot(
        data=df,
        x='combo',
        y=metric,
        hue='system',
        ci=None,
        dodge=True,
        ax=ax
    )
    ax.set_title(f'EC vs Replication: {metric_titles[metric]}')
    ax.set_xlabel('Payload Size & Virtual Users')
```

```
    ax.set_ylabel(metric_titles[metric])
    ax.tick_params(axis='x', rotation=90)

# single legend for all subplots
handles, labels = axes[0].get_legend_handles_labels()
fig.legend(handles, labels, title='System', loc='upper right')

fig.tight_layout()
plt.show()
```

```
/tmp/ipykernel_171471/1184367366.py:15: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1184367366.py:15: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1184367366.py:15: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1184367366.py:15: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```



[10]:
```
# Create grouped visualizations for bandwidth, payload size, and virtual users

# First, let's check what distinct values we have for each parameter
print(f"Unique bandwidth values: {df['bandwidth'].unique()}")
print(f"Unique payload sizes: {df['payload_size'].unique()}")
print(f"Unique virtual users: {df['virtual_user'].unique()}")
```

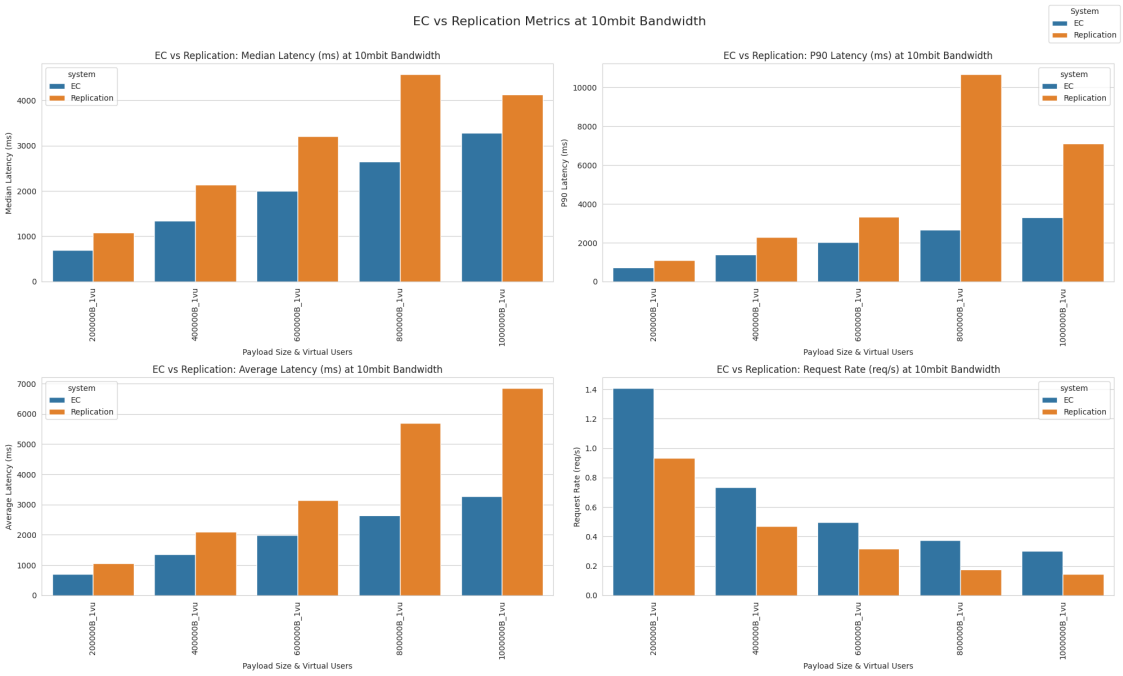
```
Unique bandwidth values: ['10mbit' '25mbit' '40mbit' '55mbit' '70mbit']
Unique payload sizes: [ 200000  400000  600000  800000 1000000]
Unique virtual users: [1]
```

[11]:
```
# 1. Grouping by bandwidth
def plot_by_bandwidth():
    # Sort bandwidths numerically using bandwidth_num
    bw_order = (
        df[['bandwidth', 'bandwidth_num']]
        .drop_duplicates()
        .sort_values('bandwidth_num')
        .bandwidth.tolist()
    )
    for bw in bw_order:
        df_bw = df[df['bandwidth'] == bw]
        if df_bw.empty:
            continue

        # set up 2 x 2 subplots for the four metrics
```

```
        fig, axes = plt.subplots(2, 2, figsize=(20, 12))
        axes = axes.flatten()
        for ax, metric in zip(axes, metrics):
            # label for x-axis
            df_bw['payload_vu'] = df_bw.apply(
                lambda row: f"{row['payload_size']}B_{row['virtual_user']}vu",␣
 ↪axis=1
            )
            sns.barplot(
                data=df_bw,
                x='payload_vu',
                y=metric,
                hue='system',
                ci=None,
                dodge=True,
                ax=ax
            )
            ax.set_title(f'EC vs Replication: {metric_titles[metric]} at {bw}␣
 ↪Bandwidth')
            ax.set_xlabel('Payload Size & Virtual Users')
            ax.set_ylabel(metric_titles[metric])
            ax.tick_params(axis='x', rotation=90)

        # single legend for all subplots
        handles, labels = axes[0].get_legend_handles_labels()
        fig.legend(handles, labels, title='System', loc='upper right')
        fig.suptitle(f'EC vs Replication Metrics at {bw} Bandwidth',␣
 ↪fontsize=16)
        fig.tight_layout(rect=[0, 0, 1, 0.97])
        plt.show()

plot_by_bandwidth()
```

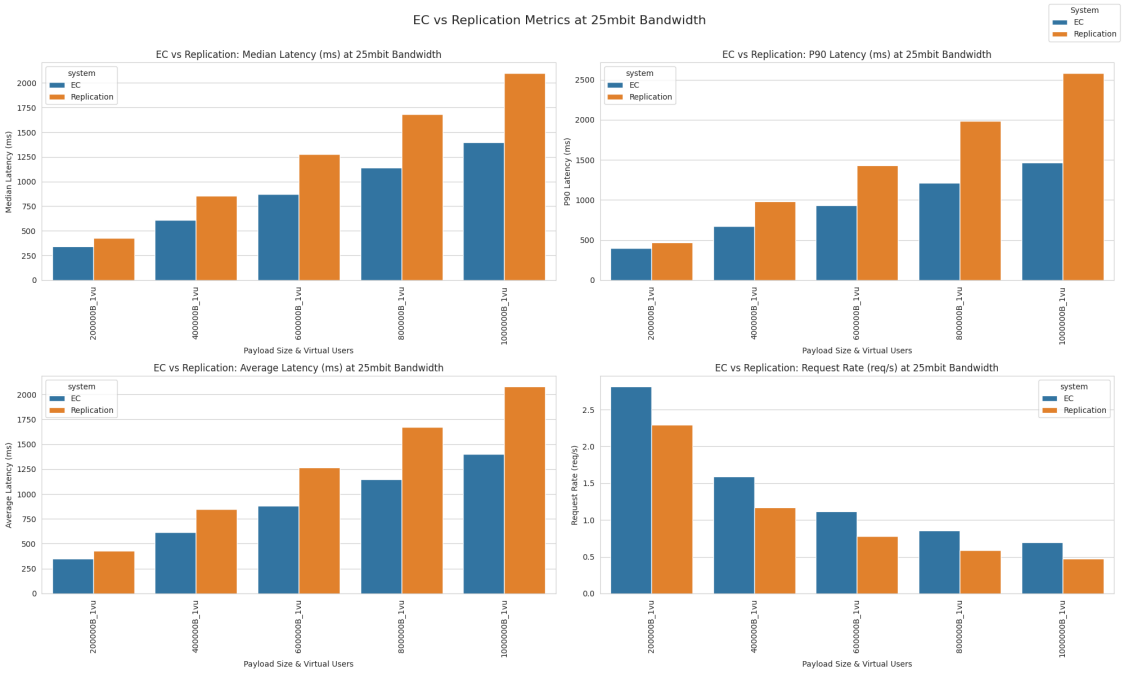
```
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```



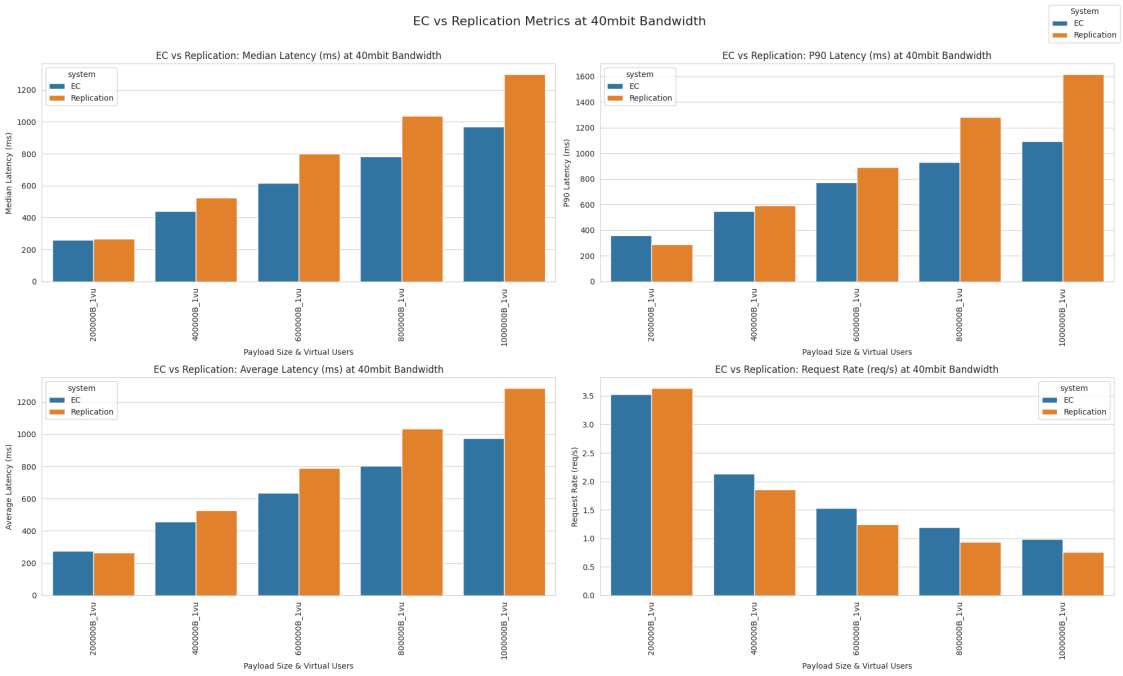

```
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```



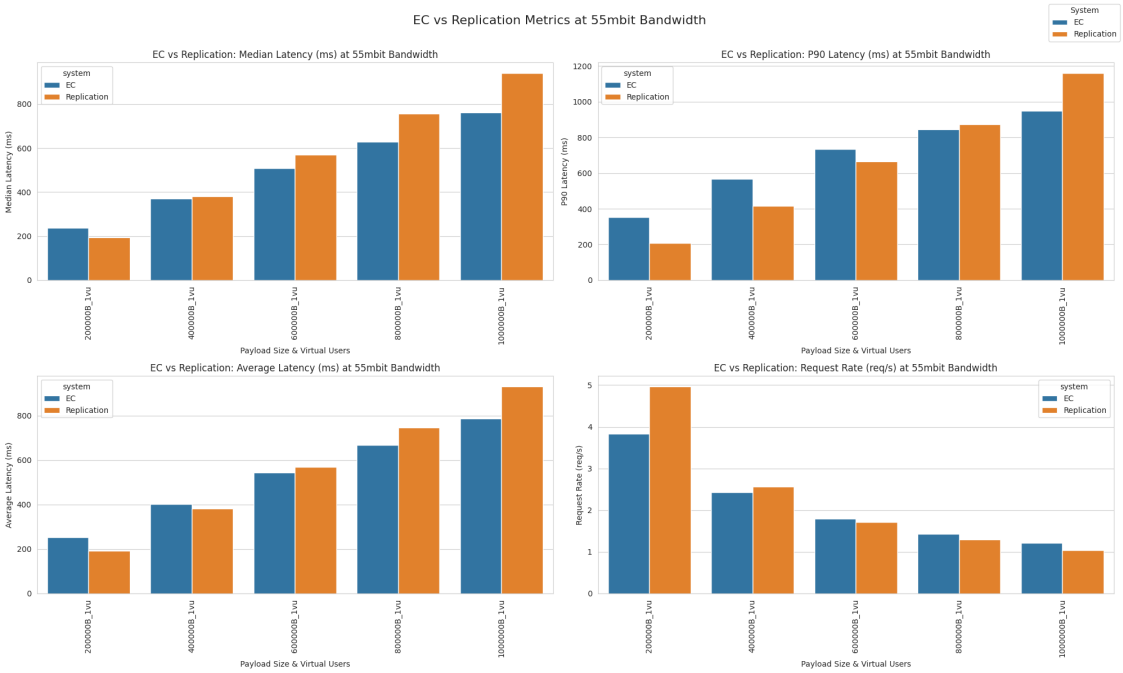

```
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```



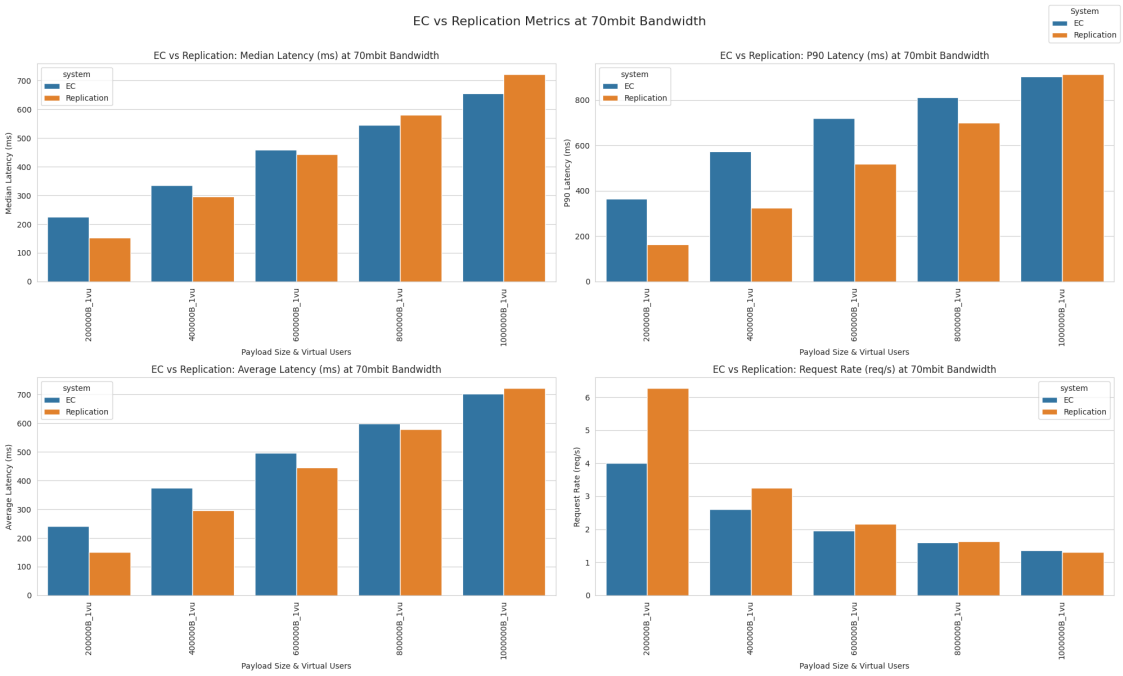

```
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```




```
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_bw['payload_vu'] = df_bw.apply(
/tmp/ipykernel_171471/2927317137.py:23: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```

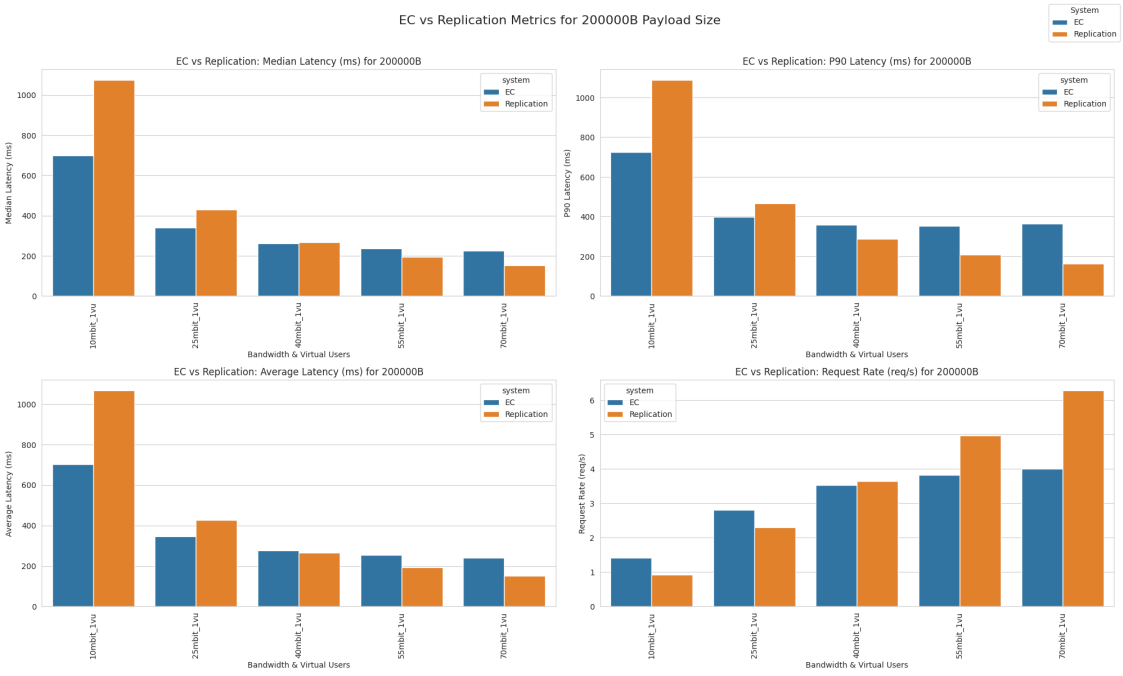


[12]:
```
# 2. Grouping by payload size (all metrics in one 2 x 2 display)
def plot_by_payload():
    payloads = df['payload_size'].unique()
    for payload in payloads:
        df_payload = df[df['payload_size'] == payload]
        if df_payload.empty:
            continue

        # Sort bandwidths numerically for this payload
        bw_order = (
            df_payload[['bandwidth', 'bandwidth_num']]
            .drop_duplicates()
            .sort_values('bandwidth_num')
            .bandwidth.tolist()
        )
        # Create combined x-axis labels and enforce ordering
        df_payload['bw_vu'] = df_payload.apply(
            lambda row: f"{row['bandwidth']}_{row['virtual_user']}vu", axis=1
        )
        vu_order = sorted(df_payload['virtual_user'].unique())
        x_order = [f"{bw}_{vu}vu" for bw in bw_order for vu in vu_order]
        df_payload['bw_vu'] = pd.Categorical(df_payload['bw_vu'],␣
 ↪categories=x_order, ordered=True)

        # set up 2 x 2 subplots for the four metrics
```

```
        fig, axes = plt.subplots(2, 2, figsize=(20, 12))
        axes = axes.flatten()
        for ax, metric in zip(axes, metrics):
            sns.barplot(
                data=df_payload,
                x='bw_vu',
                y=metric,
                hue='system',
                ci=None,
                dodge=True,
                ax=ax
            )
            ax.set_title(f"EC vs Replication: {metric_titles[metric]} for␣
 ↪{payload}B")
            ax.set_xlabel("Bandwidth & Virtual Users")
            ax.set_ylabel(metric_titles[metric])
            ax.tick_params(axis='x', rotation=90)

        # single legend for all subplots
        handles, labels = axes[0].get_legend_handles_labels()
        fig.legend(handles, labels, title='System', loc='upper right')
        fig.suptitle(f"EC vs Replication Metrics for {payload}B Payload Size",␣
 ↪fontsize=16)
        fig.tight_layout(rect=[0, 0, 1, 0.96])
        plt.show()

plot_by_payload()
```

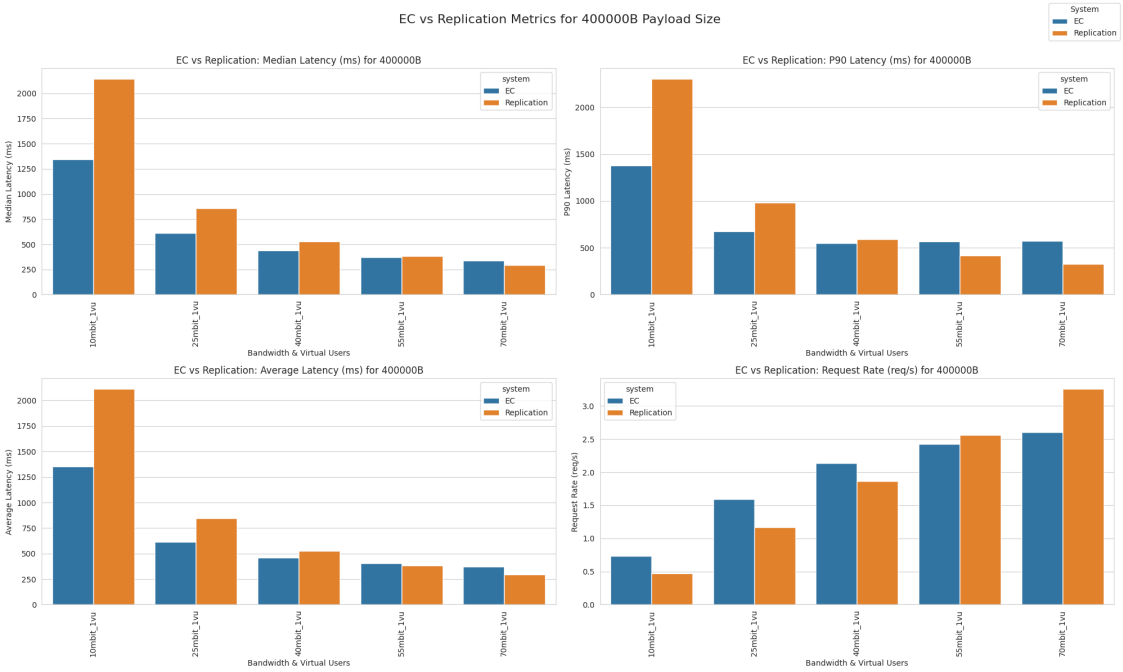
```
/tmp/ipykernel_171471/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = df_payload.apply(
/tmp/ipykernel_171471/1240405451.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = pd.Categorical(df_payload['bw_vu'], categories=x_order,
ordered=True)
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.
```

```
  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```

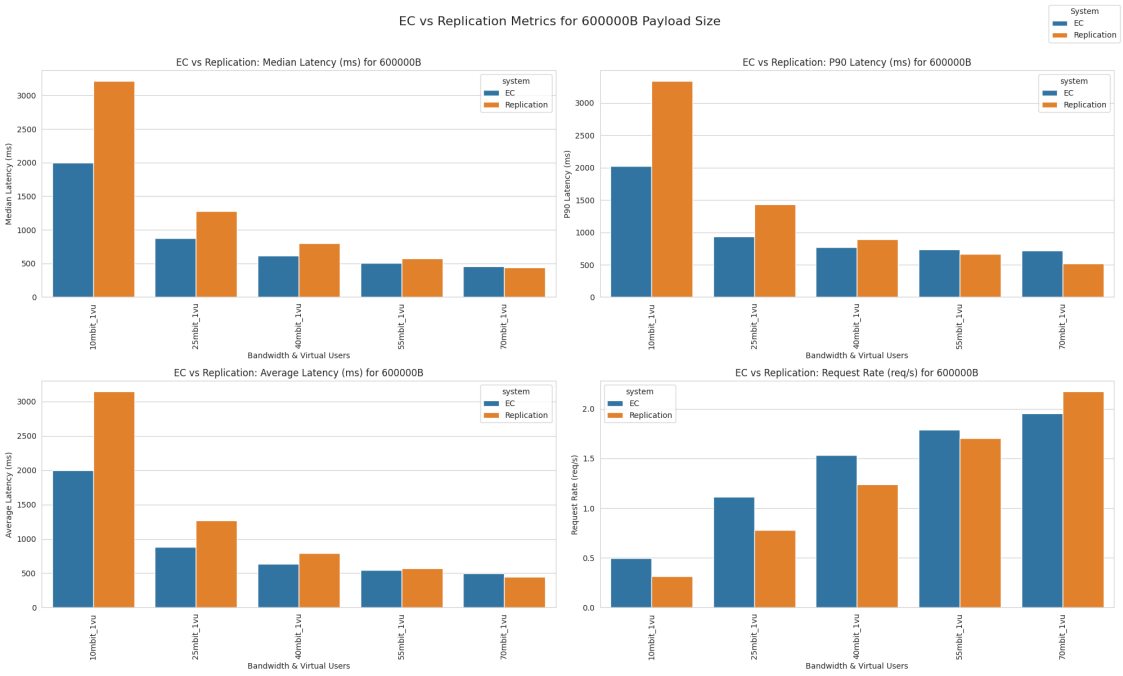



```
/tmp/ipykernel_171471/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = df_payload.apply(
/tmp/ipykernel_171471/1240405451.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = pd.Categorical(df_payload['bw_vu'], categories=x_order,
ordered=True)
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```

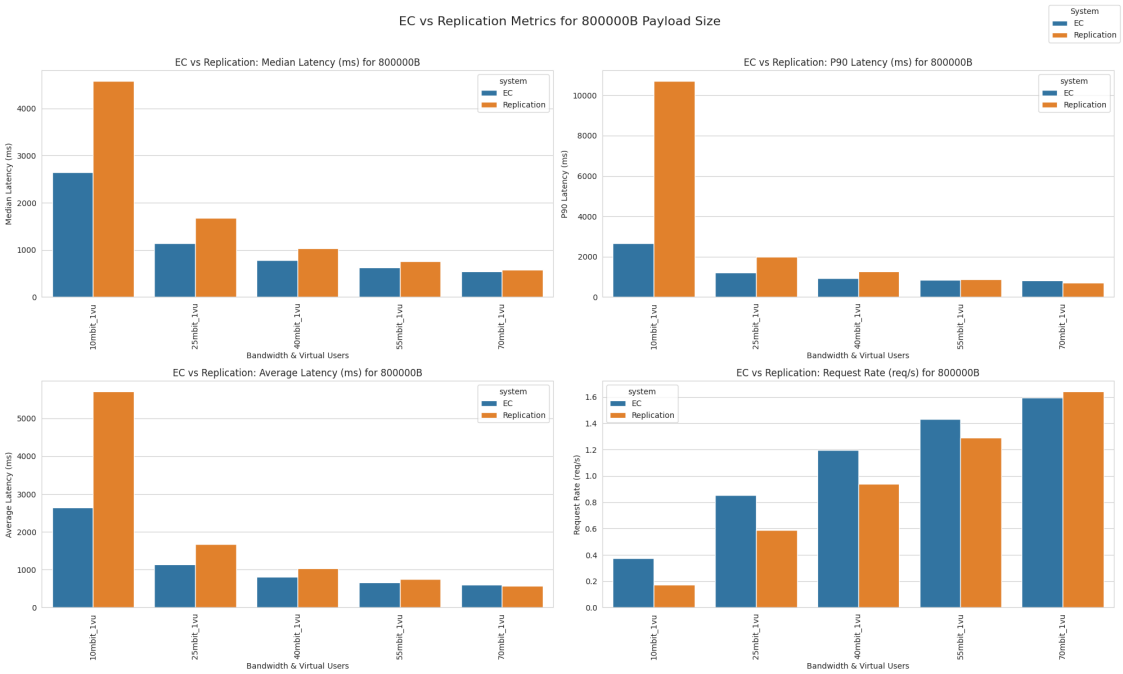


```
/tmp/ipykernel_171471/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = df_payload.apply(
/tmp/ipykernel_171471/1240405451.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = pd.Categorical(df_payload['bw_vu'], categories=x_order,
ordered=True)
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```



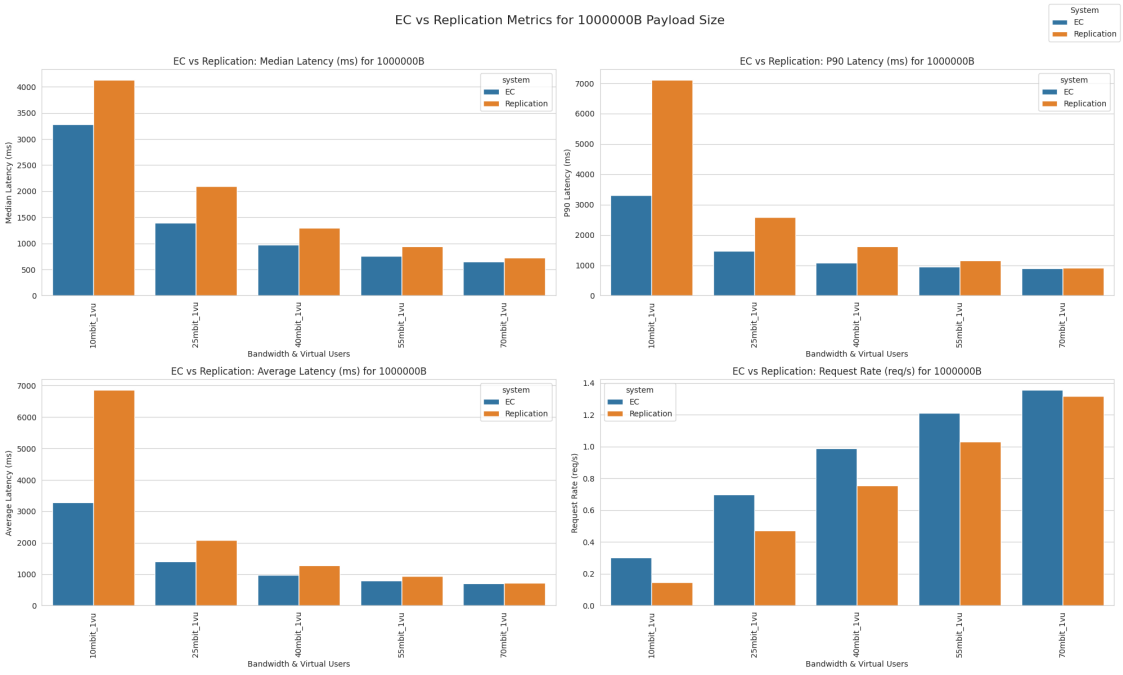

```
/tmp/ipykernel_171471/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = df_payload.apply(
/tmp/ipykernel_171471/1240405451.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = pd.Categorical(df_payload['bw_vu'], categories=x_order,
ordered=True)
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:
```

```
The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```




```
/tmp/ipykernel_171471/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = df_payload.apply(
/tmp/ipykernel_171471/1240405451.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_payload['bw_vu'] = pd.Categorical(df_payload['bw_vu'], categories=x_order,
ordered=True)
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:
```

```
The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1240405451.py:28: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```



[13]:
```
# 3. Grouping by virtual users (all metrics in one 2 x 2 display)
def plot_by_vu():
    vus = df['virtual_user'].unique()
    for vu in vus:
        df_vu = df[df['virtual_user'] == vu]
```

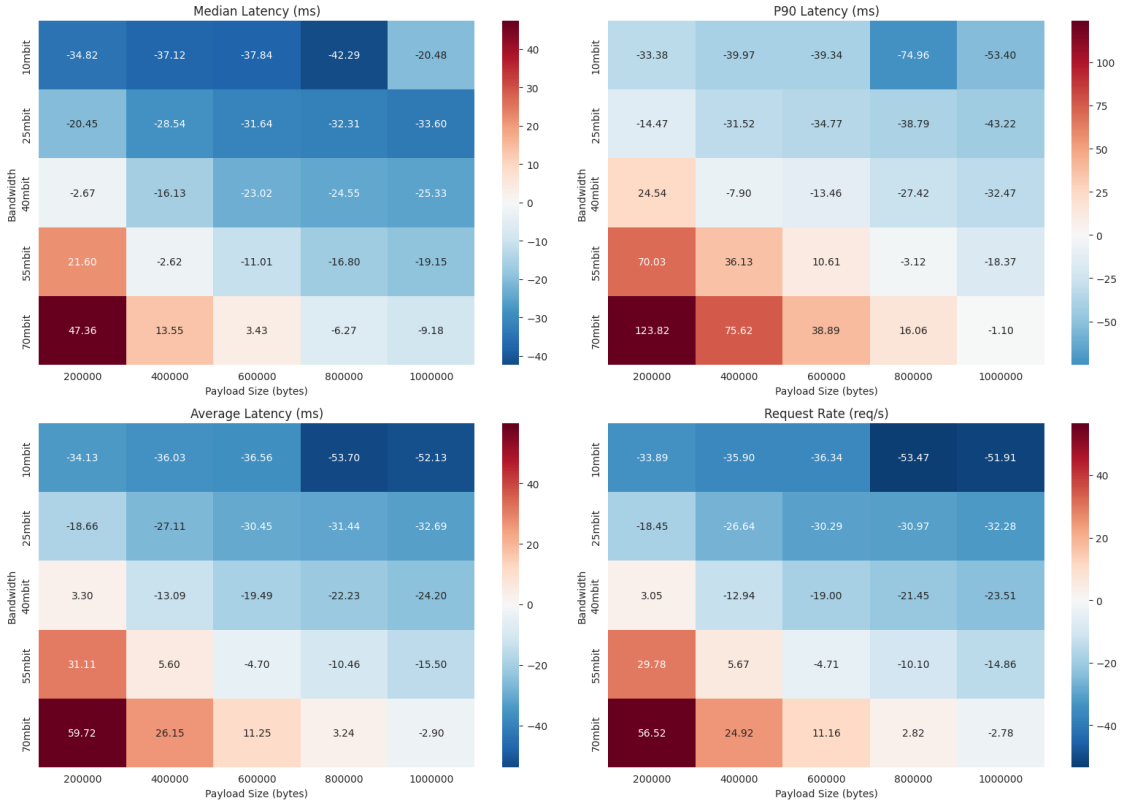
```
        if df_vu.empty:
            continue

        # set up 2 x 2 subplots for the four metrics
        fig, axes = plt.subplots(2, 2, figsize=(20, 12))
        axes = axes.flatten()

        # precompute ordering for x-axis
        bw_order = (
            df_vu[['bandwidth', 'bandwidth_num']]
            .drop_duplicates()
            .sort_values('bandwidth_num')
            .bandwidth.tolist()
        )
        ps_order = sorted(df_vu['payload_size'].unique())
        x_order = [f"{bw}_{ps}B" for bw in bw_order for ps in ps_order]

        # apply categorical ordering
        df_vu['bw_payload'] = df_vu.apply(
            lambda row: f"{row['bandwidth']}_{row['payload_size']}B", axis=1
        )
        df_vu['bw_payload'] = pd.Categorical(df_vu['bw_payload'],␣
↪categories=x_order, ordered=True)

        for ax, metric in zip(axes, metrics):
            sns.barplot(
                data=df_vu,
                x='bw_payload',
                y=metric,
                hue='system',
                ci=None,
                dodge=True,
                ax=ax
            )
            ax.set_title(f"EC vs Replication: {metric_titles[metric]} (VU:␣
↪{vu})")
            ax.set_xlabel('Bandwidth & Payload Size')
            ax.set_ylabel(metric_titles[metric])
            ax.tick_params(axis='x', rotation=90)

        # single legend for all subplots
        handles, labels = axes[0].get_legend_handles_labels()
        fig.legend(handles, labels, title='System', loc='upper right')
        fig.suptitle(f"EC vs Replication Metrics for {vu} Virtual Users",␣
↪fontsize=16)
        fig.tight_layout(rect=[0, 0, 1, 0.96])
        plt.show()
```

```
plot_by_vu()
```

```
/tmp/ipykernel_171471/1253657513.py:30: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1253657513.py:30: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1253657513.py:30: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
/tmp/ipykernel_171471/1253657513.py:30: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(
```

[14]:
```
# 4. Create heatmaps to visualize the performance difference between EC and␣
 ↪Replication
# This can help identify which configurations benefit most from EC vs␣
 ↪Replication

def plot_heatmaps_by_vu():
    df_ec = df[df['system'] == 'EC'].copy()
    df_repl = df[df['system'] == 'Replication'].copy()
    common_conditions = pd.merge(
        df_ec[['payload_size','virtual_user','bandwidth','bandwidth_num']],
        df_repl[['payload_size','virtual_user','bandwidth','bandwidth_num']],
        on=['payload_size','virtual_user','bandwidth','bandwidth_num']
    )
    if common_conditions.empty:
        return

    df_ec_f = pd.merge(df_ec, common_conditions,␣
 ↪on=['payload_size','virtual_user','bandwidth','bandwidth_num'])
    df_repl_f = pd.merge(df_repl, common_conditions,␣
 ↪on=['payload_size','virtual_user','bandwidth','bandwidth_num'])

    for vu in sorted(common_conditions['virtual_user'].unique()):
        fig, axes = plt.subplots(2, 2, figsize=(16, 12))
        axes = axes.flatten()
        for ax, metric in zip(axes, metrics):
            diff_data = []
            df_ec_vu = df_ec_f[df_ec_f['virtual_user'] == vu]
            df_repl_vu = df_repl_f[df_repl_f['virtual_user'] == vu]
            for _, ec_row in df_ec_vu.iterrows():
                repl_row = df_repl_vu[
                    (df_repl_vu['payload_size'] == ec_row['payload_size']) &
                    (df_repl_vu['bandwidth'] == ec_row['bandwidth'])
                ].iloc[0]

                if metric in ['med', 'p90', 'avg']:
                    denom = repl_row[metric] if repl_row[metric] != 0 else 1
                    diff = (ec_row[metric] - repl_row[metric]) / denom * 100
                else:
                    denom = ec_row[metric] if ec_row[metric] != 0 else 1
                    diff = (repl_row[metric] - ec_row[metric]) / denom * 100

                diff_data.append({
                    'bandwidth': ec_row['bandwidth'],
                    'bandwidth_num': ec_row['bandwidth_num'],
                    'payload_size': ec_row['payload_size'],
                    'diff': diff
                })
```

```
            diff_df = pd.DataFrame(diff_data)
            bw_order = (
                diff_df[['bandwidth','bandwidth_num']]
                .drop_duplicates()
                .sort_values('bandwidth_num')
                ['bandwidth']
                .tolist()
            )
            pivot = diff_df.pivot(index='bandwidth', columns='payload_size',␣
 ↪values='diff')\
                           .reindex(bw_order)

            sns.heatmap(pivot, annot=True, cmap='RdBu_r', center=0, fmt='.2f',␣
 ↪ax=ax)
            ax.set_title(metric_titles[metric])
            ax.set_xlabel('Payload Size (bytes)')
            ax.set_ylabel('Bandwidth')

        fig.suptitle(f'Performance Difference Heatmaps (VU: {vu})', fontsize=16)
        plt.tight_layout(rect=[0,0,1,0.96])
        plt.show()

plot_heatmaps_by_vu()
```


Performance Difference Heatmaps (VU: 1)
Median Latency (ms)
10mbit
25mbit
40mbit
55mbit
70mbit
Bandwidth
-34.82 -37.12 -37.84 -42.29 -20.48
-20.45 -28.54 -31.64 -32.31 -33.60
-2.67 -16.13 -23.02 -24.55 -25.33
21.60 -2.62 -11.01 -16.80 -19.15
47.36 13.55 3.43 -6.27 -9.18
200000 400000 600000 800000 1000000
Payload Size (bytes)
40 30 20 10 0 −10 −20 −30 −40
P90 Latency (ms)
-33.38 -39.97 -39.34 -74.96 -53.40
-14.47 -31.52 -34.77 -38.79 -43.22
24.54 -7.90 -13.46 -27.42 -32.47
70.03 36.13 10.61 -3.12 -18.37
123.82 75.62 38.89 16.06 -1.10
100 75 50 25 0 −25 −50
Average Latency (ms)
-34.13 -36.03 -36.56 -53.70 -52.13
-18.66 -27.11 -30.45 -31.44 -32.69
3.30 -13.09 -19.49 -22.23 -24.20
31.11 5.60 -4.70 -10.46 -15.50
59.72 26.15 11.25 3.24 -2.90
40 20 0 −20 −40
Request Rate (req/s)
-33.89 -35.90 -36.34 -53.47 -51.91
-18.45 -26.64 -30.29 -30.97 -32.28
3.05 -12.94 -19.00 -21.45 -23.51
29.78 5.67 -4.71 -10.10 -14.86
56.52 24.92 11.16 2.82 -2.78
40 20 0 −20 −40