benchmark_k6_read

July 13, 2025

[25]:
```
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 = "/read_avgnet/"

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

# Helper to extract data from files

# 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'_read_(.+?)_(\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
```

[25]:

	system	payload_size	virtual_user	bandwidth	med	p90	\
0	EC	200000	1	10mbit	9.124901	10.055491	
1	Replication	200000	1	10mbit	2.551200	3.139400	
2	EC	200000	1	25mbit	8.878701	10.085901	
3	Replication	200000	1	25mbit	2.214601	2.711220	
4	EC	200000	1	40mbit	8.708501	9.745851	
5	Replication	200000	1	40mbit	2.206701	2.701601	
6	EC	200000	1	55mbit	8.527501	9.542181	
7	Replication	200000	1	55mbit	2.203500	2.690460	
8	EC	200000	1	70mbit	8.179301	9.215161	
9	Replication	200000	1	70mbit	2.144900	2.622800	
10	EC	400000	1	10mbit	12.772001	16.404281	
11	Replication	400000	1	10mbit	3.159750	4.113140	
12	EC	400000	1	25mbit	12.520901	15.096402	
13	Replication	400000	1	25mbit	2.842400	3.647481	
14	EC	400000	1	40mbit	12.175551	13.843451	
15	Replication	400000	1	40mbit	2.858900	3.801000	
16	EC	400000	1	55mbit	11.962101	13.686821	
17	Replication	400000	1	55mbit	2.866700	3.818521	
18	EC	400000	1	70mbit	11.404601	12.973821	
19	Replication	400000	1	70mbit	2.743200	3.581411	
20	EC	600000	1	10mbit	15.724501	17.259242	
21	Replication	600000	1	10mbit	3.845950	4.728531	
22	EC	600000	1	25mbit	15.408301	17.127521	
23	Replication	600000	1	25mbit	3.260801	3.969001	
24	EC	600000	1	40mbit	15.225651	16.935872	
25	Replication	600000	1	40mbit	3.250301	3.964220	
26	EC	600000	1	55mbit	14.982252	16.750471	
27	Replication	600000	1	55mbit	3.226001	3.908161	
28	EC	600000	1	70mbit	14.340301	16.054062	
29	Replication	600000	1	70mbit	3.196500	3.886910	
30	EC	800000	1	10mbit	19.107251	21.944972	
31	Replication	800000	1	10mbit	4.211951	5.957451	
32	EC	800000	1	25mbit	18.645702	21.561252	

33	Replication	800000	1	25mbit	3.962301	6.422200
34	EC	800000	1	40mbit	18.405901	20.948841
35	Replication	800000	1	40mbit	3.885200	5.902941
36	EC	800000	1	55mbit	18.267102	22.380782
37	Replication	800000	1	55mbit	3.894900	6.102000
38	EC	800000	1	70mbit	17.485401	21.007102
39	Replication	800000	1	70mbit	3.812800	5.708961
40	EC	1000000	1	10mbit	22.146502	25.057702
41	Replication	1000000	1	10mbit	4.604700	6.251681
42	EC	1000000	1	25mbit	21.368901	24.712341
43	Replication	1000000	1	25mbit	4.214101	5.506330
44	EC	1000000	1	40mbit	21.237251	24.414312
45	Replication	1000000	1	40mbit	4.214251	5.480670
46	EC	1000000	1	55mbit	20.939201	23.878402
47	Replication	1000000	1	55mbit	4.298851	6.445801
48	EC	1000000	1	70mbit	20.203501	23.103322
49	Replication	1000000	1	70mbit	4.151101	5.356040

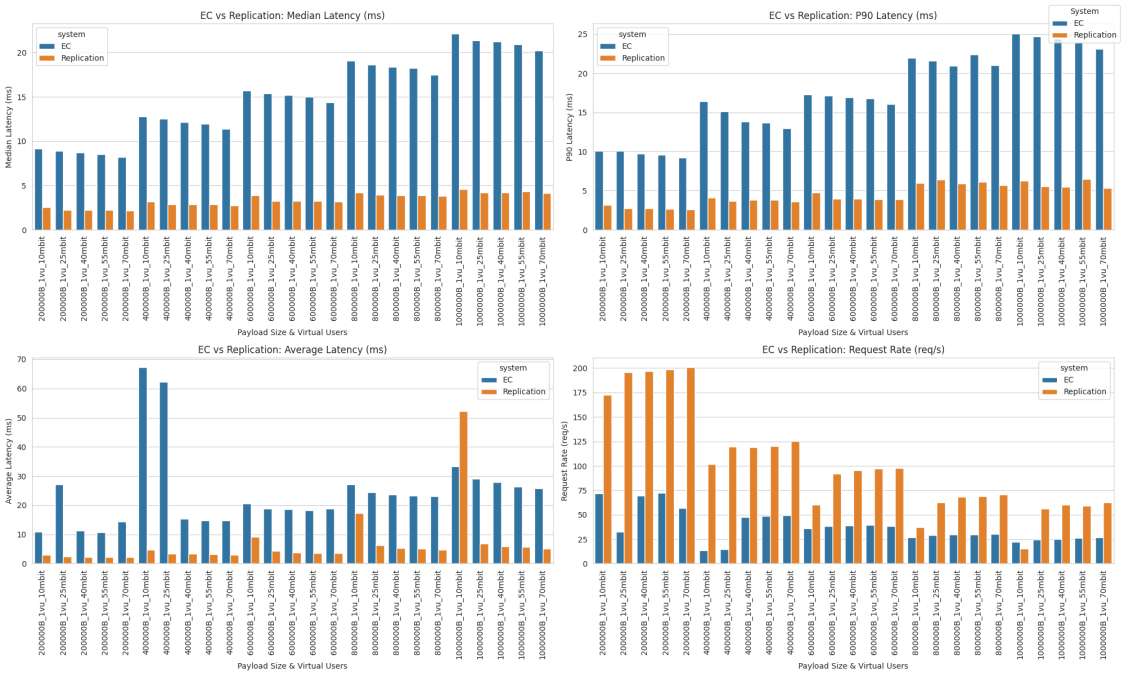
	avg	rate	bandwidth_num	combo
0	10.860901	71.560057	10000	200000B_1vu_10mbit
1	2.985204	172.481319	10000	200000B_1vu_10mbit
2	27.179235	32.630497	25000	200000B_1vu_25mbit
3	2.354469	195.552406	25000	200000B_1vu_25mbit
4	11.303761	69.505690	40000	200000B_1vu_40mbit
5	2.305885	197.046616	40000	200000B_1vu_40mbit
6	10.762659	72.170053	55000	200000B_1vu_55mbit
7	2.293517	198.336889	55000	200000B_1vu_55mbit
8	14.392082	56.709423	70000	200000B_1vu_70mbit
9	2.230576	200.676736	70000	200000B_1vu_70mbit
10	67.232828	13.505693	10000	400000B_1vu_10mbit
11	4.684858	101.690860	10000	400000B_1vu_10mbit
12	62.172990	14.587000	25000	400000B_1vu_25mbit
13	3.395078	119.415664	25000	400000B_1vu_25mbit
14	15.406891	47.675817	40000	400000B_1vu_40mbit
15	3.287103	119.124967	40000	400000B_1vu_40mbit
16	14.826273	48.919919	55000	400000B_1vu_55mbit
17	3.187288	120.100178	55000	400000B_1vu_55mbit
18	14.857108	49.250975	70000	400000B_1vu_70mbit
19	2.989706	125.451981	70000	400000B_1vu_70mbit
20	20.531078	36.227728	10000	600000B_1vu_10mbit
21	9.238988	60.157541	10000	600000B_1vu_10mbit
22	18.904356	38.453432	25000	600000B_1vu_25mbit
23	4.245174	91.746041	25000	600000B_1vu_25mbit
24	18.595093	38.904588	40000	600000B_1vu_40mbit
25	3.846926	95.439353	40000	600000B_1vu_40mbit
26	18.285675	39.244829	55000	600000B_1vu_55mbit
27	3.627947	97.371060	55000	600000B_1vu_55mbit

28	18.862731	38.363590	70000	600000B_1vu_70mbit
29	3.511334	97.896048	70000	600000B_1vu_70mbit
30	27.075902	26.900681	10000	800000B_1vu_10mbit
31	17.297442	37.215449	10000	800000B_1vu_10mbit
32	24.437212	29.233778	25000	800000B_1vu_25mbit
33	6.324088	62.585153	25000	800000B_1vu_25mbit
34	23.716280	29.908257	40000	800000B_1vu_40mbit
35	5.348494	68.060497	40000	800000B_1vu_40mbit
36	23.316528	29.731001	55000	800000B_1vu_55mbit
37	5.087205	68.867878	55000	800000B_1vu_55mbit
38	22.989338	30.217363	70000	800000B_1vu_70mbit
39	4.749964	70.600655	70000	800000B_1vu_70mbit
40	33.255462	22.189536	10000	1000000B_1vu_10mbit
41	52.175193	15.402991	10000	1000000B_1vu_10mbit
42	29.058840	24.496383	25000	1000000B_1vu_25mbit
43	6.916651	56.240790	25000	1000000B_1vu_25mbit
44	27.944875	25.200413	40000	1000000B_1vu_40mbit
45	5.903930	60.001215	40000	1000000B_1vu_40mbit
46	26.359037	26.376127	55000	1000000B_1vu_55mbit
47	5.739029	59.012397	55000	1000000B_1vu_55mbit
48	25.816443	26.766462	70000	1000000B_1vu_70mbit
49	5.126644	62.710817	70000	1000000B_1vu_70mbit

[26]:
```
# 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_288815/1184367366.py:15: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1184367366.py:15: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1184367366.py:15: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1184367366.py:15: FutureWarning:

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

  sns.barplot(
```



[27]:
```
# 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()}")
```

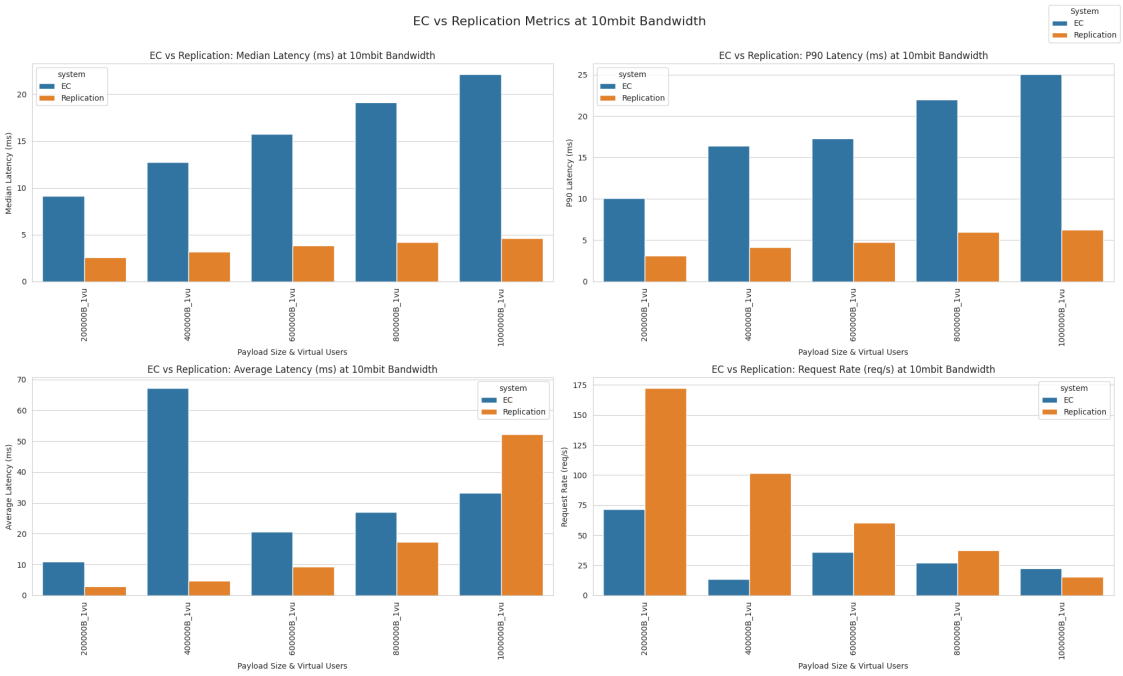
```
Unique bandwidth values: ['10mbit' '25mbit' '40mbit' '55mbit' '70mbit']
Unique payload sizes: [ 200000  400000  600000  800000 1000000]
Unique virtual users: [1]
```

[28]:
```
# 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()
```

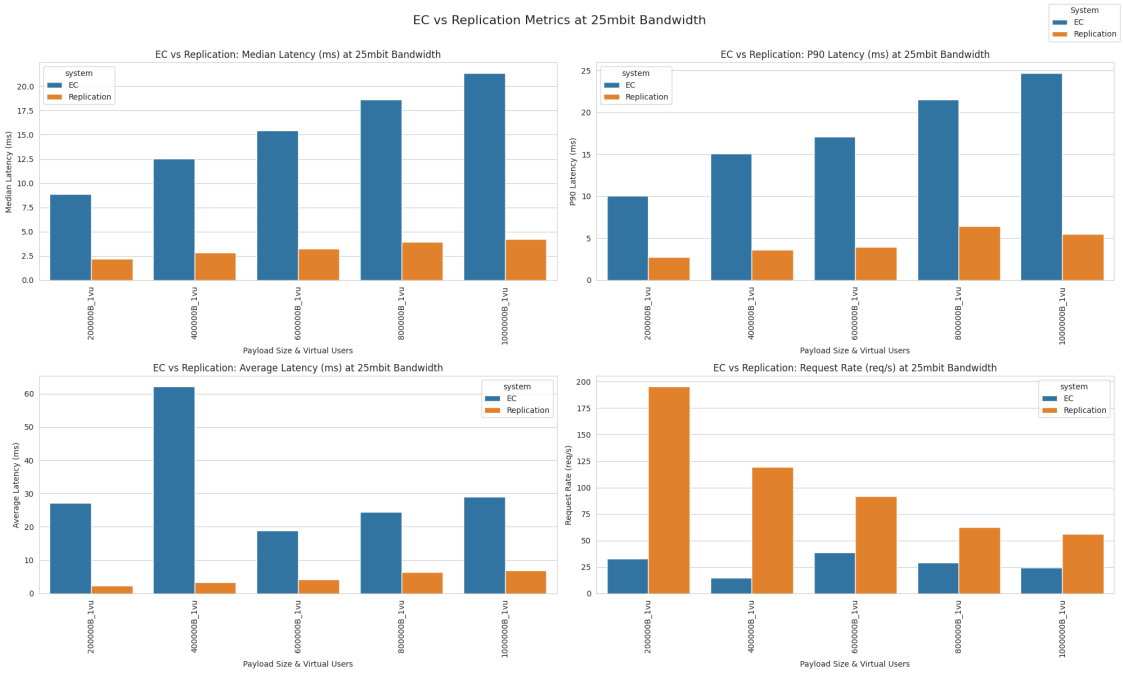
```
/tmp/ipykernel_288815/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
```



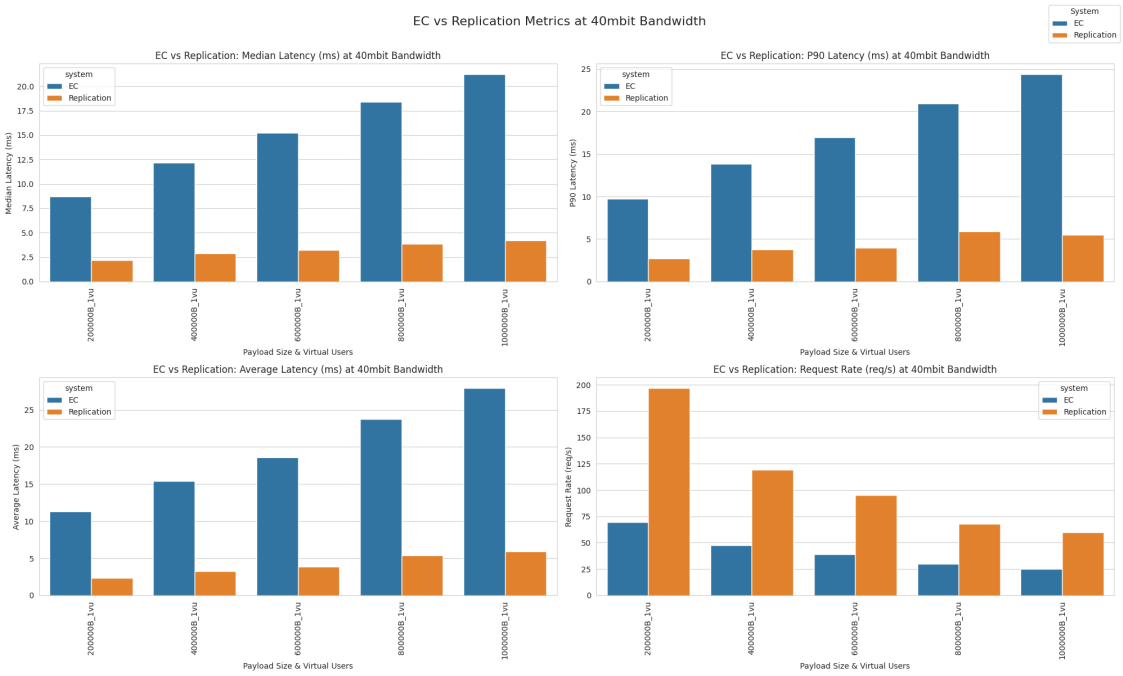

```
/tmp/ipykernel_288815/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
```



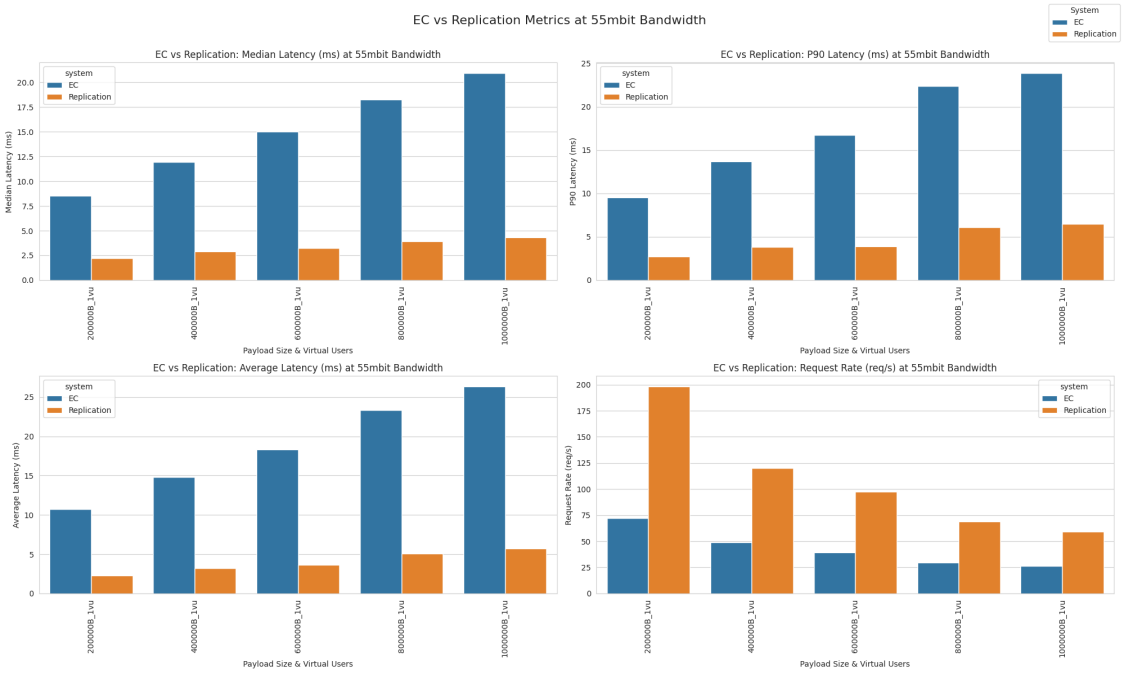

```
/tmp/ipykernel_288815/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
```



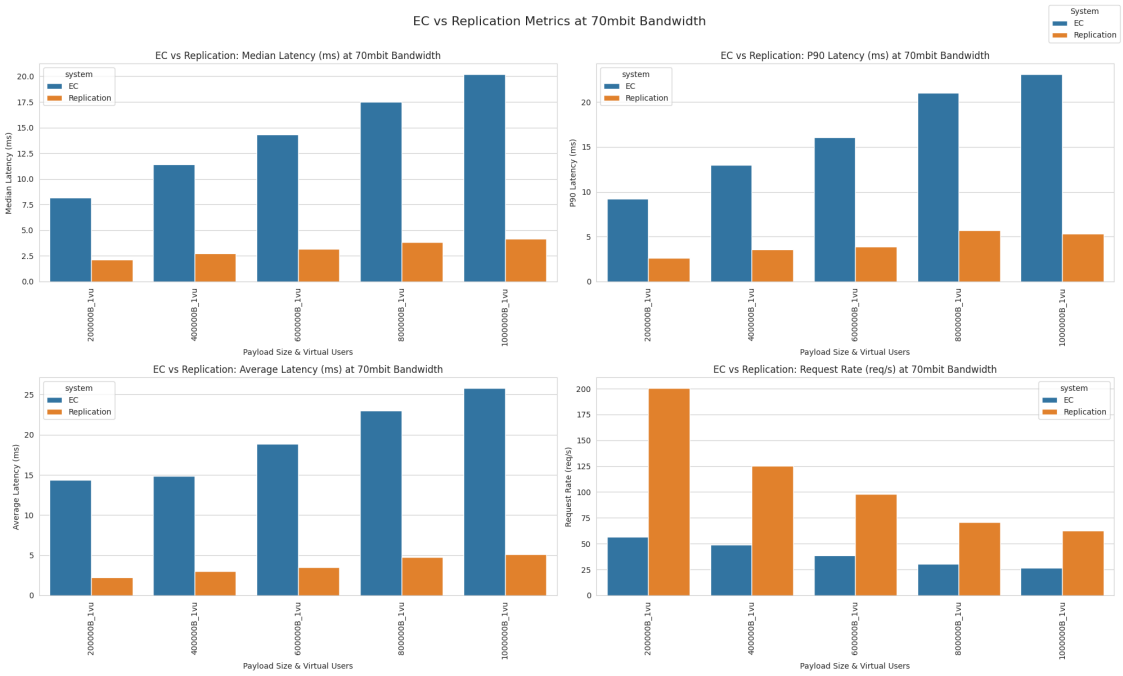

```
/tmp/ipykernel_288815/2927317137.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
```




```
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/2927317137.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_288815/2927317137.py:23: FutureWarning:

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

  sns.barplot(
```

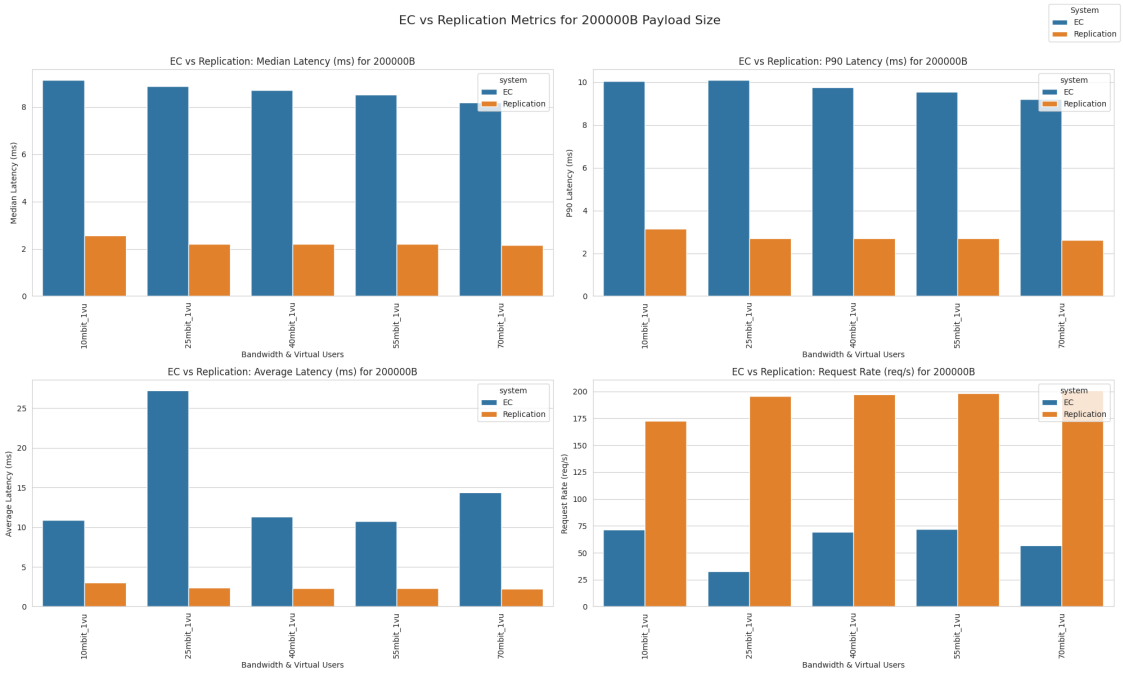


[29]:
```
# 2. Grouping by payload size (all metrics in one 2 x 2 display)
def plot_by_payload():
    payloads = df['payload_size'].unique()
    for payload in payloads:
        df_payload = df[df['payload_size'] == payload]
        if df_payload.empty:
            continue

        # 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()
```

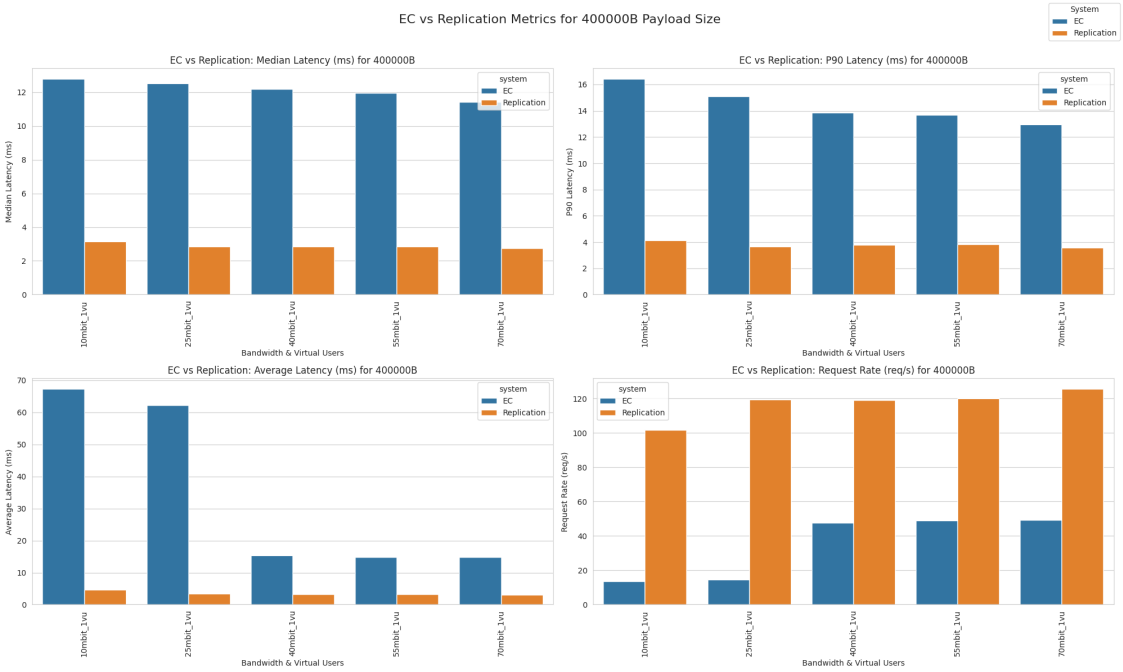
```
/tmp/ipykernel_288815/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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_288815/1240405451.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_288815/1240405451.py:28: FutureWarning:

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

```
  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
```

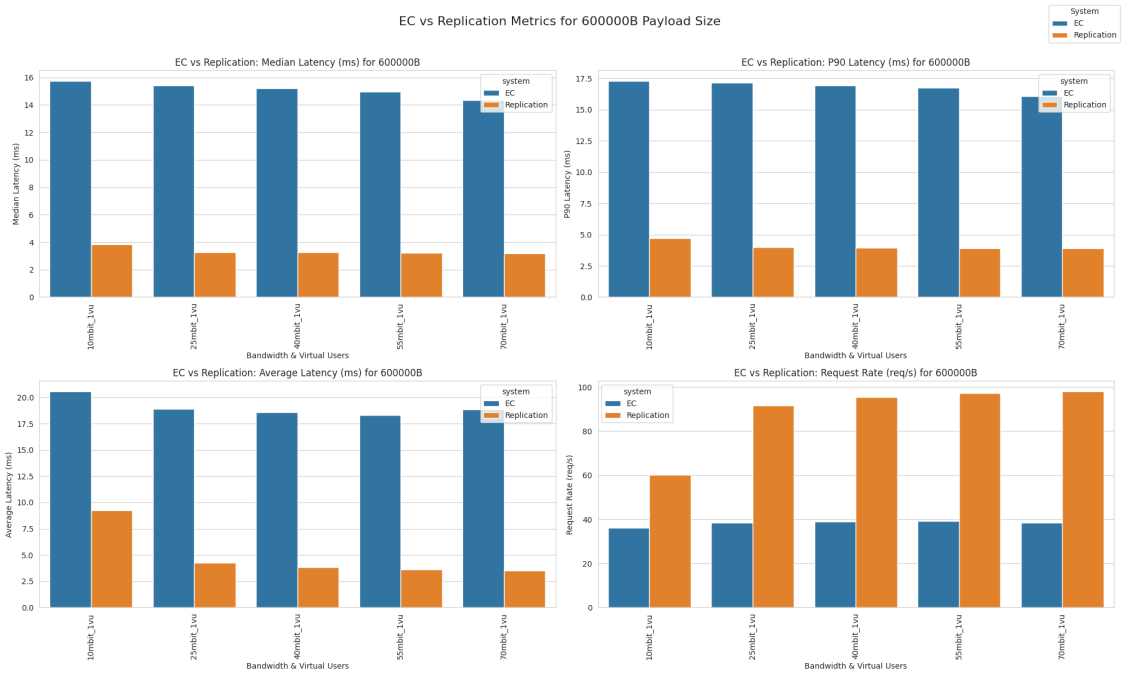



```
/tmp/ipykernel_288815/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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_288815/1240405451.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_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
```

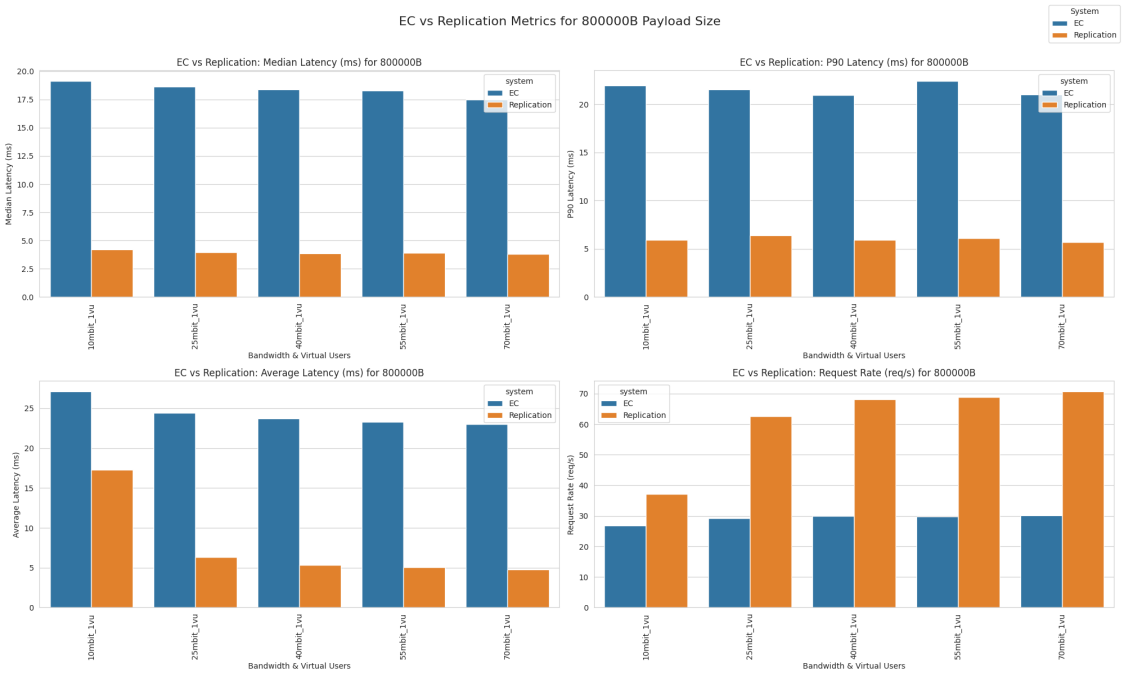


```
/tmp/ipykernel_288815/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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_288815/1240405451.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_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
```



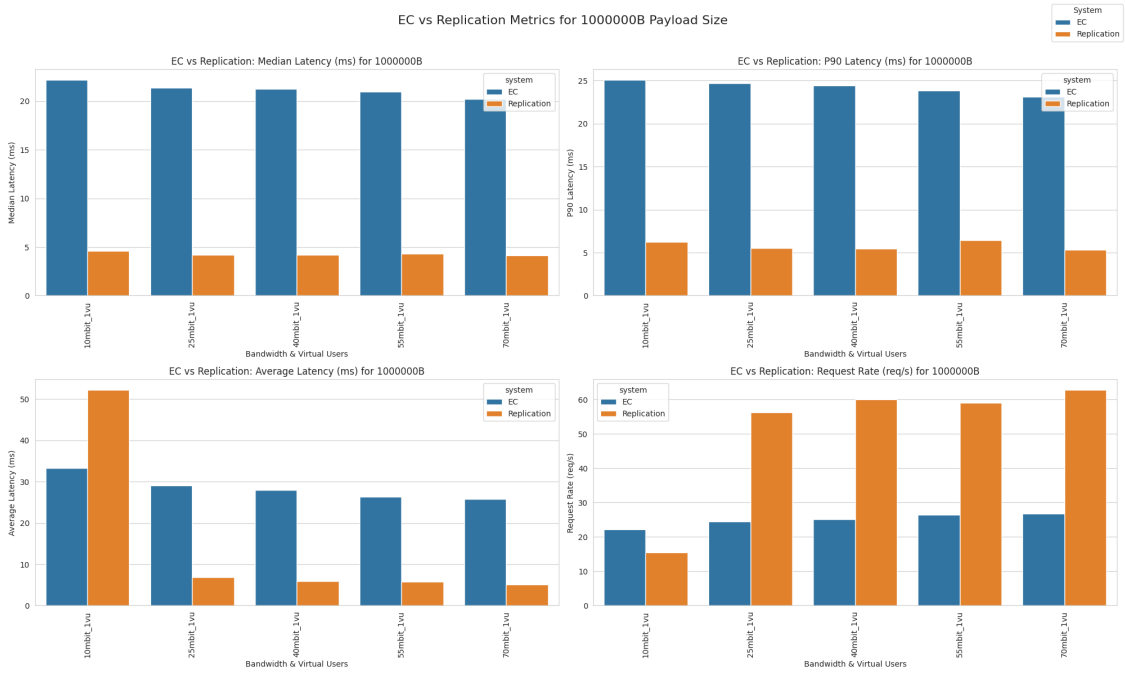

```
/tmp/ipykernel_288815/1240405451.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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_288815/1240405451.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_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:
```

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
```




```
/tmp/ipykernel_288815/1240405451.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_288815/1240405451.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_288815/1240405451.py:28: FutureWarning:
```

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1240405451.py:28: FutureWarning:

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

  sns.barplot(
```

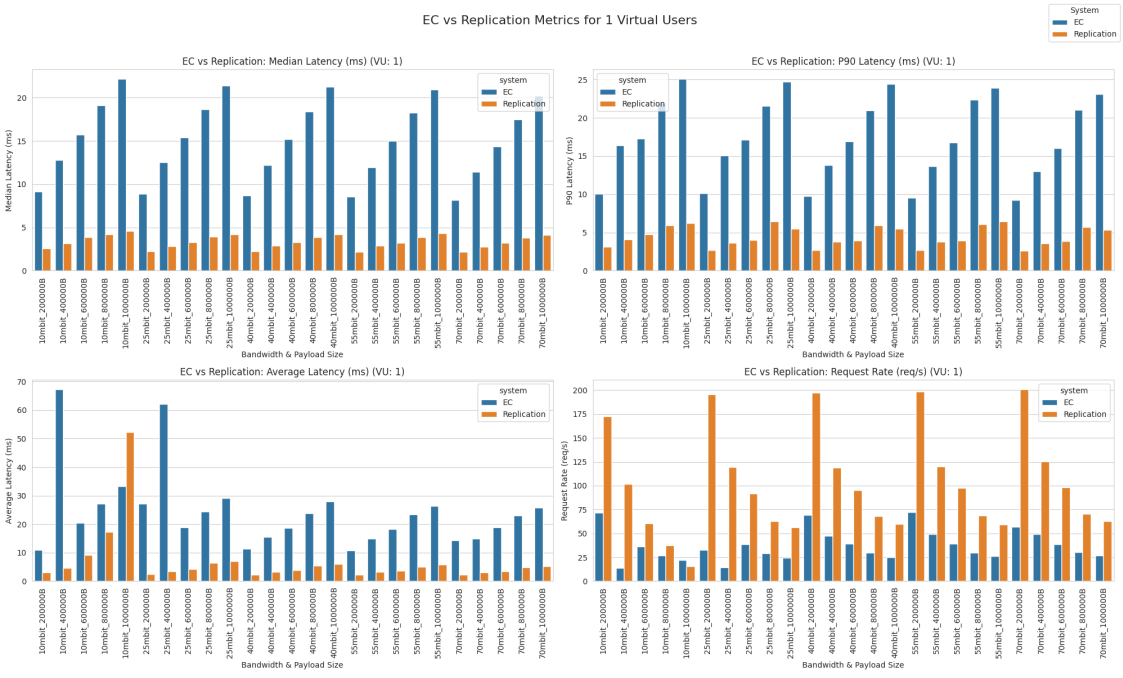


[30]:
```
# 3. Grouping by virtual users (all metrics in one 2 x 2 display)
def plot_by_vu():
    vus = df['virtual_user'].unique()
    for vu in vus:
        df_vu = df[df['virtual_user'] == vu]
```

```
        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()
```

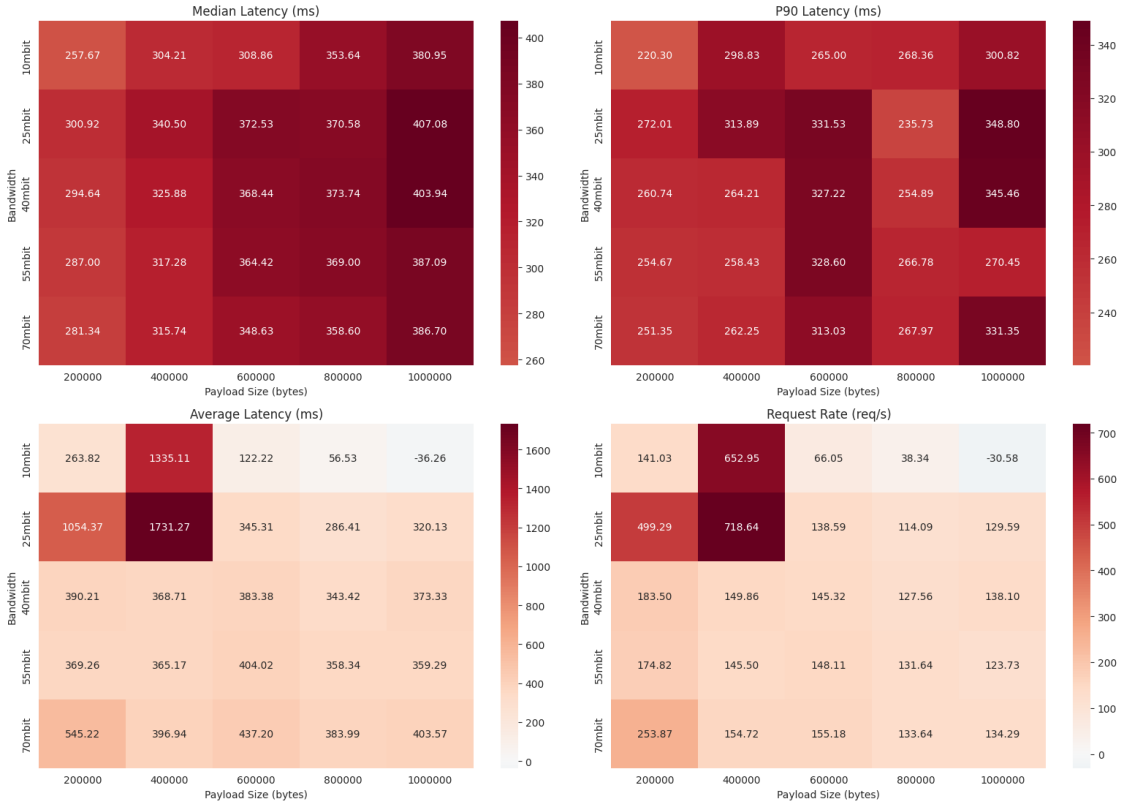
```
plot_by_vu()
```

```
/tmp/ipykernel_288815/1253657513.py:30: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1253657513.py:30: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1253657513.py:30: FutureWarning:

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

  sns.barplot(
/tmp/ipykernel_288815/1253657513.py:30: FutureWarning:

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

  sns.barplot(
```



[31]:
```
# 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
257.67 304.21 308.86 353.64 380.95
300.92 340.50 372.53 370.58 407.08
294.64 325.88 368.44 373.74 403.94
287.00 317.28 364.42 369.00 387.09
281.34 315.74 348.63 358.60 386.70
200000 400000 600000 800000 1000000
Payload Size (bytes)
400 380 360 340 320 300 280 260
P90 Latency (ms)
220.30 298.83 265.00 268.36 300.82
272.01 313.89 331.53 235.73 348.80
260.74 264.21 327.22 254.89 345.46
254.67 258.43 328.60 266.78 270.45
251.35 262.25 313.03 267.97 331.35
340 320 300 280 260 240
Average Latency (ms)
263.82 1335.11 122.22 56.53 -36.26
1054.37 1731.27 345.31 286.41 320.13
390.21 368.71 383.38 343.42 373.33
369.26 365.17 404.02 358.34 359.29
545.22 396.94 437.20 383.99 403.57
1600 1400 1200 1000 800 600 400 200 0
Request Rate (req/s)
141.03 652.95 66.05 38.34 -30.58
499.29 718.64 138.59 114.09 129.59
183.50 149.86 145.32 127.56 138.10
174.82 145.50 148.11 131.64 123.73
253.87 154.72 155.18 133.64 134.29
700 600 500 400 300 200 100 0