In this notebook, we will process data on CPU and Memory usage obtained from monitoring the nodes.

In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
pd.set_option('max_columns', None)
```

In [2]:
```
# VIRT: Amount of virtual memory used by the process.
# RES: Amount of resident memory used by the process.
# SHR: Amount of shared memory used by the process.
# %CPU: The share of CPU time used by the process since the last update.
# %MEM: The share of physical memory used.

def getDataframeFromTopOutput(df):
    df['timestamp'] = pd.to_datetime(df[0].str.strip('['))
    df[['VIRT','RES','SHR','CPU','MEM']] = df[1].str.split(' ', expand=True)[[5,6,7,9,10]].astype(float)
    df = df.drop([0,1],axis=1)
    return df
```

In [3]:
```
def getAverageByMinute(df):
    df1 = df.copy().drop(['timestamp'],axis=1)
    # convert to datetime
    df1['datetime'] = pd.to_datetime(df['timestamp'])
    df1 = df1.set_index('datetime')
    # get average in every minute
    df1 = df1.resample('1T').mean()
    df1 = df1.reset_index()
    df1 = df1.rename(columns={"datetime": "timestamp"})
    return df1
```

1205_top_ipfs.txt:

```
[2022-12-05 00:00:01] 17286 sixiao 20 0 1502240 185076 45136 S 0.0 1.1 1:09.70 ipfs
[2022-12-05 00:00:11] 17286 sixiao 20 0 1502240 185076 45136 S 0.0 1.1 1:10.11 ipfs
[2022-12-05 00:00:21] 17286 sixiao 20 0 1502240 185076 45136 S 0.0 1.1 1:10.56 ipfs
[2022-12-05 00:00:31] 17286 sixiao 20 0 1502240 185076 45136 S 0.0 1.1 1:10.92 ipfs
[2022-12-05 00:00:41] 17286 sixiao 20 0 1502240 185076 45136 S 6.7 1.1 1:11.73 ipfs
...
```

In [4]:
```
df1 = pd.read_csv('1205_top_ipfs.txt', sep=']', header=None)
df1 = getDataframeFromTopOutput(df1)
df1 = getAverageByMinute(df1)
df1['node'] = 'ipfs'

df1.shape
```

Out[4]: (1440, 7)

In [5]:
```
df1.describe()
```

Out[5]:

	VIRT	RES	SHR	CPU	MEM
count	1.440000e+03	1440.000000	1440.000000	1440.000000	1440.000000
mean	1.561475e+06	227492.439352	45429.019444	3.943611	1.380856
std	6.314195e+04	20274.600643	111.837993	3.642878	0.125761
min	1.502240e+06	185076.000000	45136.000000	0.000000	1.100000
25%	1.503456e+06	210200.000000	45392.000000	1.116667	1.300000
50%	1.572044e+06	224404.333333	45456.000000	3.266667	1.400000
75%	1.646192e+06	247554.000000	45556.000000	5.566667	1.500000
max	1.646704e+06	281976.000000	45556.000000	29.783333	1.700000

1205_top_swarm.txt:

```
[2022-12-05 00:00:01] 15233 bee 20 0 944920 292548 29608 R 0.0 1.8 0:30.61 bee
[2022-12-05 00:00:11] 15233 bee 20 0 944920 292548 29608 S 0.0 1.8 0:30.99 bee
[2022-12-05 00:00:21] 15233 bee 20 0 944920 292548 29608 S 13.3 1.8 0:31.38 bee
[2022-12-05 00:00:31] 15233 bee 20 0 944920 298980 29672 S 0.0 1.8 0:31.94 bee
[2022-12-05 00:00:41] 15233 bee 20 0 944920 305376 29672 S 0.0 1.9 0:32.35 bee
...
```

In [6]:
```
df2 = pd.read_csv('1205_top_swarm.txt', sep=']', header=None)

df2 = getDataframeFromTopOutput(df2)
df2 = getAverageByMinute(df2)
df2['node'] = 'swarm'

df2.shape
```

Out[6]: (1440, 7)

In [7]:
```
df2.describe()
```

Out[7]:

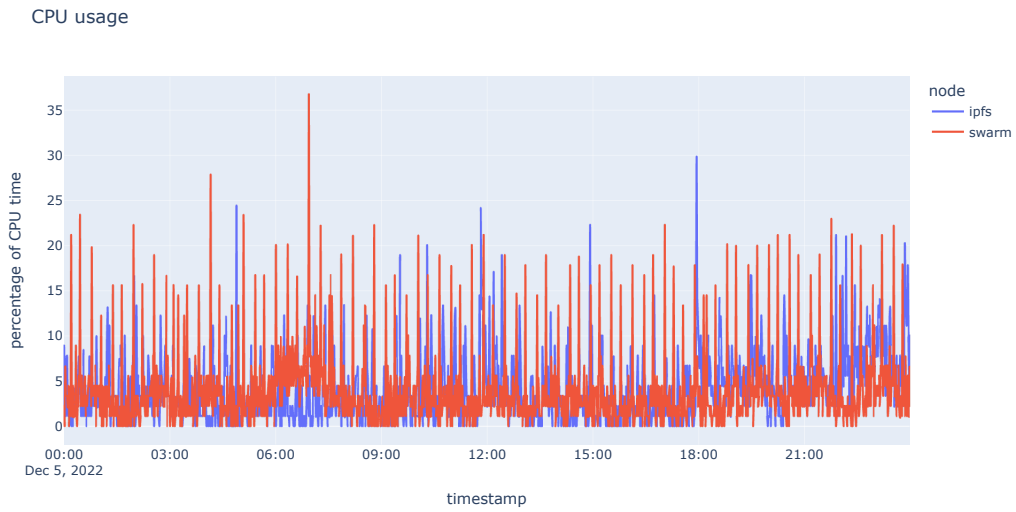
	VIRT	RES	SHR	CPU	MEM
count	1.440000e+03	1440.000000	1440.000000	1440.000000	1440.000000
mean	1.084167e+06	328056.194444	29845.057870	3.820648	1.995949
std	4.630390e+03	24445.494041	34.968674	4.198211	0.148428
min	9.450053e+05	279516.000000	29640.000000	0.000000	1.700000
25%	1.084456e+06	301890.333333	29860.000000	1.116667	1.833333
50%	1.084456e+06	335296.333333	29860.000000	2.233333	2.033333
75%	1.084456e+06	350384.000000	29860.000000	4.450000	2.100000
max	1.084456e+06	373856.000000	29860.000000	36.683333	2.300000

In [8]:
```
df = pd.concat([df1, df2])
df.head()
```

Out[8]:

	timestamp	VIRT	RES	SHR	CPU	MEM	node
0	2022-12-05 00:00:00	1.502240e+06	185076.000000	45136.000000	1.116667	1.1	ipfs
1	2022-12-05 00:01:00	1.502240e+06	197458.000000	45178.666667	8.883333	1.2	ipfs
2	2022-12-05 00:02:00	1.502283e+06	210810.666667	45200.000000	3.350000	1.3	ipfs
3	2022-12-05 00:03:00	1.502304e+06	211952.000000	45200.000000	3.266667	1.3	ipfs
4	2022-12-05 00:04:00	1.502368e+06	210327.333333	45200.000000	1.116667	1.3	ipfs

In [9]:
```
fig = px.line(df, x='timestamp', y='CPU', color='node', title='CPU usage')
fig.update_yaxes(title="percentage of CPU time")
fig.show()
```



In [10]:
```
fig = px.line(df, x='timestamp', y='MEM', color='node', title='Memory usage')
fig.update_yaxes(title="percentage of total memory")
fig.show()
```