In [10]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [11]:

```
## Azure
# vm_table_data_link = "https://azurepublicdatasettraces.blob.core.windows.n
# vm_table_data = pd.read_csv(vm_table_data_link, compression = "gzip")
vm_table_data = pd.read_csv("vm_table_data_azure.csv")
vm_table_data = vm_table_data[['vm_id', 'vm_creation_timestamp', 'vm_deleti
                               'avg_cpu', 'p95_max_cpu', 'vm_category', 'vm_
vm_table_data.sort_values(by=['vm_creation_timestamp', 'vm_deletion_timestam
vm_table_data['start_hour'] = vm_table_data['vm_creation_timestamp'] // 3600
vm_table_data['end_hour'] = vm_table_data['vm_deletion_timestamp'] // 3600
vm_table_data
```

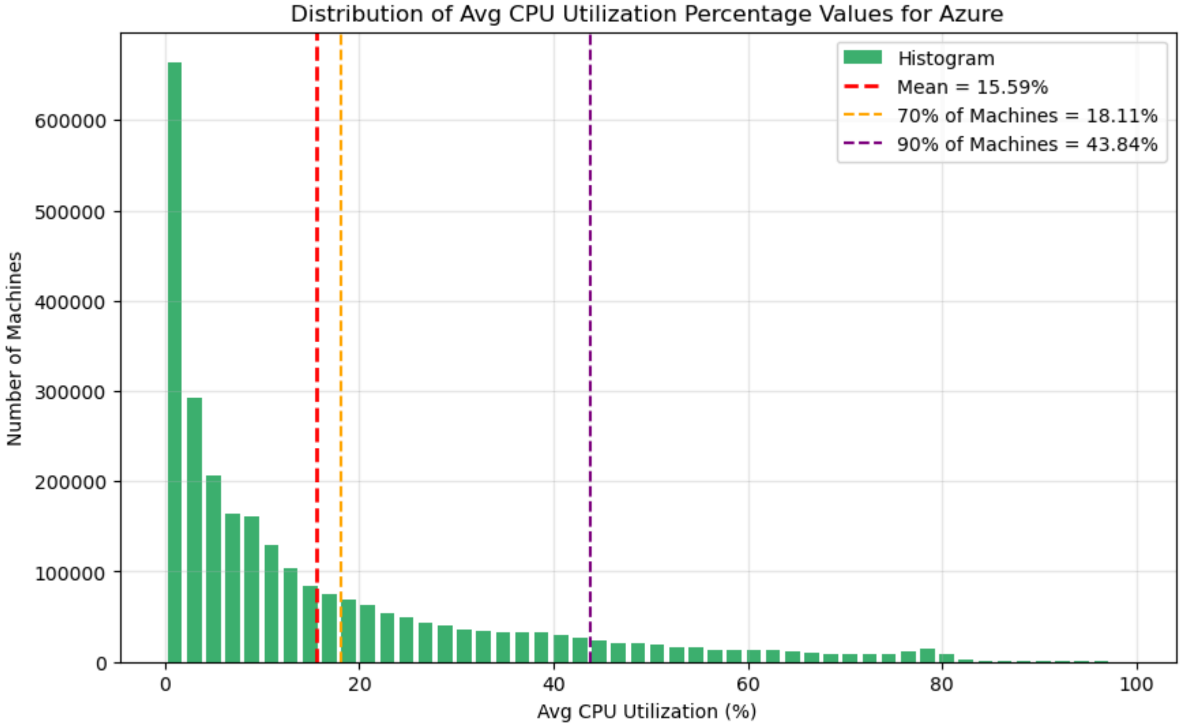
Out[11]:

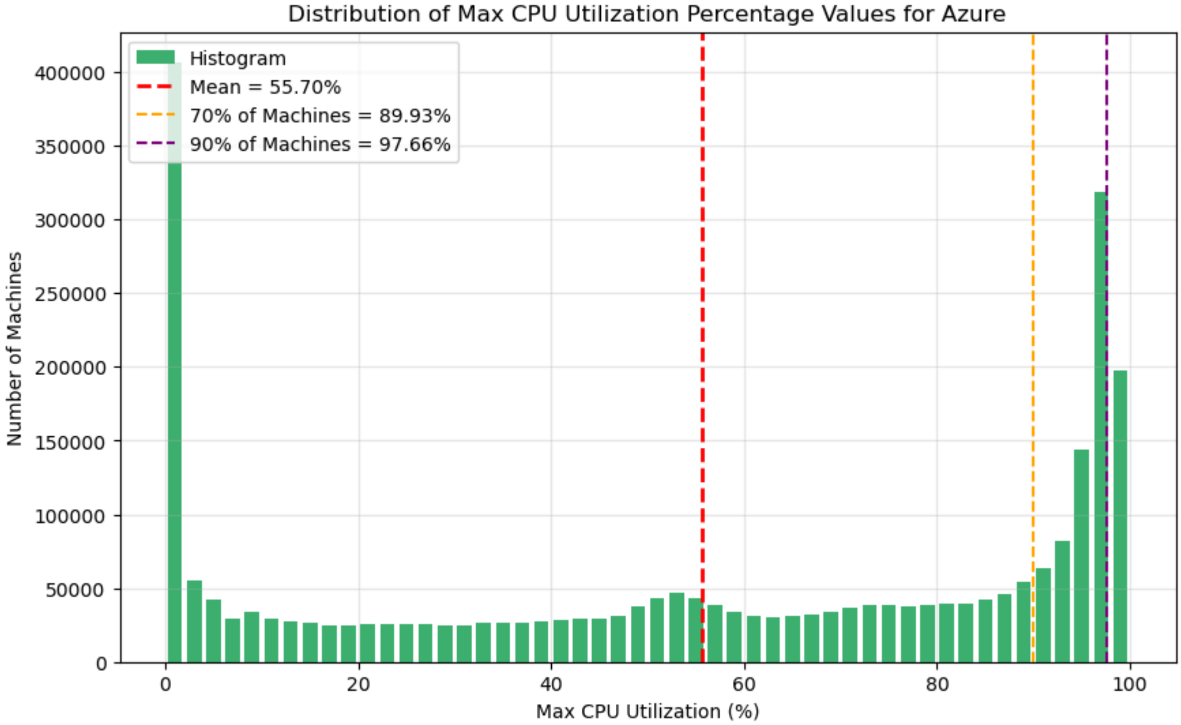
	vm_id	vm_creation_timestamp
0	rKggHO/04j31UFy65mDTwtjdMQL/G03xWfl3xGeiilB4/W...	424500
1	YrR8gPtBmfNaOdnNEW5If1SdTqQgGQHEnLHGPjySt53bKW...	1133100
2	xzQ++JF1UAkh70CDhmzkiOo+DQn+E2TLErCFKEmSswv1pl...	0
3	vZEivnhabRmImDr+JqKqZnpIM3WxtypwoxjfjnklR/idyR...	228300
4	MqvcZ6Au5ouI6if56MJHmoSqHtX8oRv0dPkaxCId3aUcr1...	1395600
...	...	...
2695542	CfZn37rcUvC4sVjWik6yIutOzNfno3c4dg6eIoqpaSE8P8...	141300
2695543	D5jsQPZSlO+KakH/yp7bPV5hrKPhyxMrh0WAzMVarUDFpf...	0
2695544	FP9Lf4/jjWgWI9HS80x1NoeFwOPhLQo1ACPgqjtBF3+z9L...	1744800
2695545	thW1eyboLMZJy6GgeClpLllRfqn0q7JgDYarlC9Jm5tg6i...	387000
2695546	TqMw/UmeYGTWCvWBdL+yIw6+Pz3Vzj/OgIZNepu3scPY94...	1300500

2695547 rows × 11 columns

In [12]:

```
def plot_resource_analysis(dataframe, col_name, xlabel, title):
    plt.figure(figsize=(10, 6))
    data = dataframe[col_name].dropna()
    mean_val = data.mean()
    p70 = np.percentile(data, 70)
    p90 = np.percentile(data, 90)

    color = "dodgerblue" if col_name.startswith('cpu') else "mediumseagreen'
    plt.hist(data, bins=50, rwidth=0.75, color=color, label='Histogram')
    plt.title(title)
    plt.xlabel(f"{xlabel} Utilization (%)")
    plt.ylabel("Number of Machines")
    plt.axvline(mean_val, color='red', linestyle='--', linewidth=2, label=f
    plt.axvline(p70, color='orange', linestyle='--', linewidth=1.5, label=f
    plt.axvline(p90, color='purple', linestyle='--', linewidth=1.5, label=f
    plt.grid(True, alpha=0.3)
    plt.legend()
    plt.savefig(f'azure_{col_name}_distribution.png', dpi=300, bbox_inches=
    plt.show()

plot_resource_analysis(vm_table_data, 'avg_cpu', "Avg CPU", "Distribution of
plot_resource_analysis(vm_table_data, 'max_cpu', "Max CPU", "Distribution of
```





```
In [13]: hourly_data = (vm_table_data.assign(hour=lambda df: df.apply(lambda row: ran
         hourly_data
```

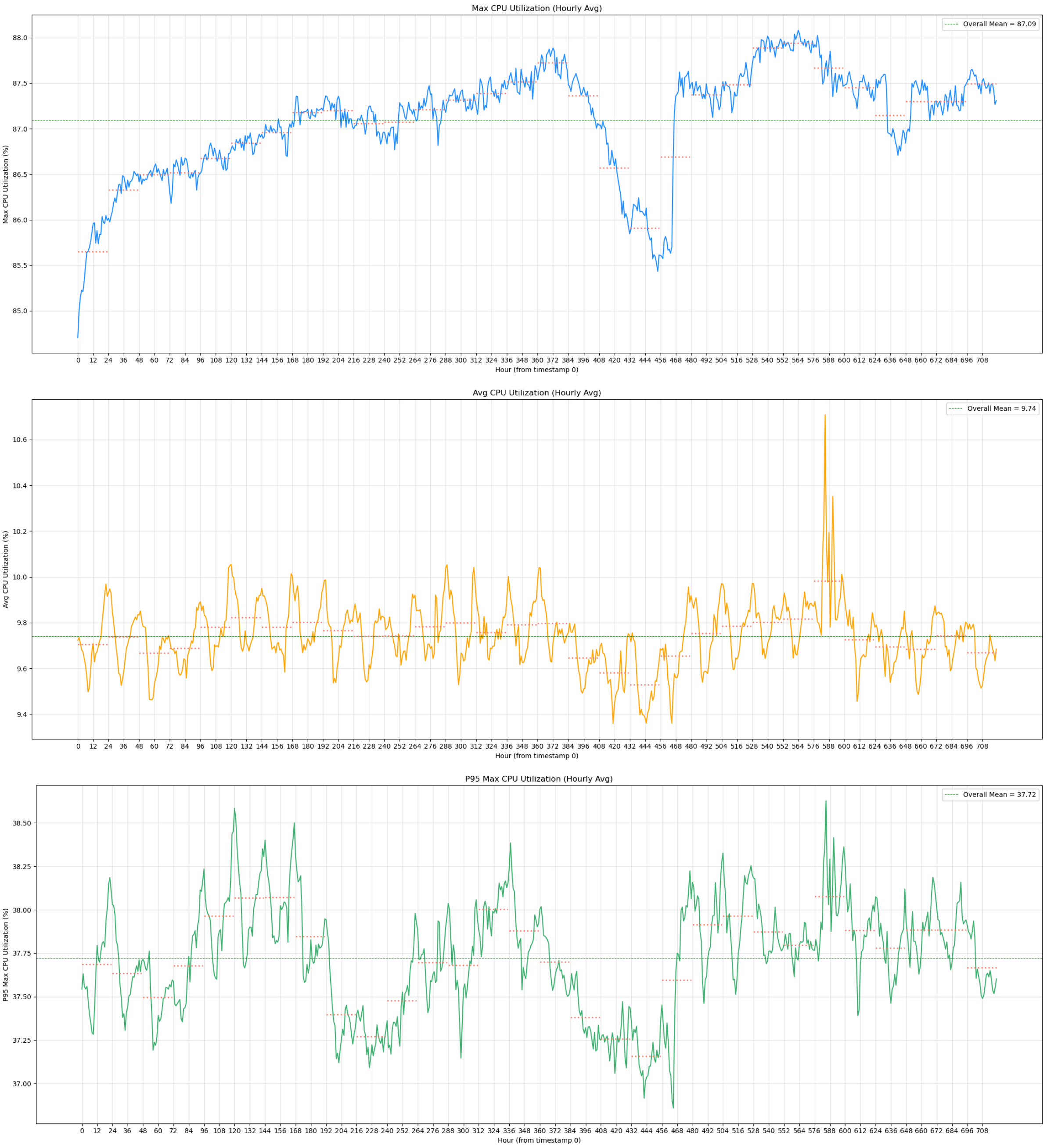
Out[13]:

	vm_id	vm_creation_timestamp
0	rKggHO/04j31UFy65mDTwtjdMQL/G03xWfl3xGeiilB4/W...	424500
0	rKggHO/04j31UFy65mDTwtjdMQL/G03xWfl3xGeiilB4/W...	424500
1	YrR8gPtBmfNaOdnNEW5If1SdTqQgGQHEnLHGPjySt53bKW...	1133100
2	xzQ++JF1UAkh70CDhmzkiOo+DQn+E2TLErCFKEmSswv1pl...	0
2	xzQ++JF1UAkh70CDhmzkiOo+DQn+E2TLErCFKEmSswv1pl...	0
...	...	...
2695544	FP9Lf4/jjWgWI9HS80x1NoeFwOPhLQo1ACPgqjtBF3+z9L...	1744800
2695544	FP9Lf4/jjWgWI9HS80x1NoeFwOPhLQo1ACPgqjtBF3+z9L...	1744800
2695544	FP9Lf4/jjWgWI9HS80x1NoeFwOPhLQo1ACPgqjtBF3+z9L...	1744800
2695545	thW1eyboLMZJy6GgeClpLlIRfqn0q7JgDYarlC9Jm5tg6i...	387000
2695546	TqMw/UmeYGTWCvWBdL+ylw6+Pz3Vzj/OgIZNepu3scPY94...	1300500

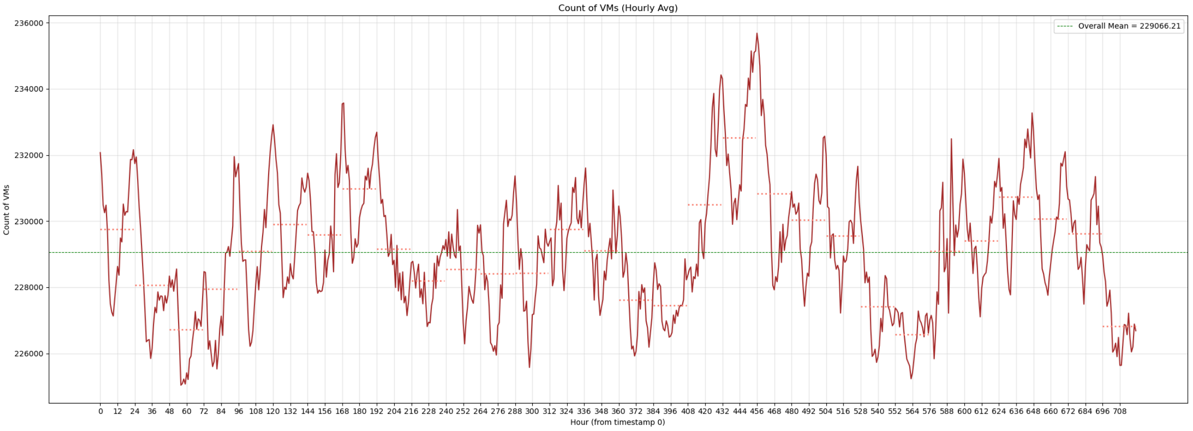
164927669 rows × 12 columns

In [15]:

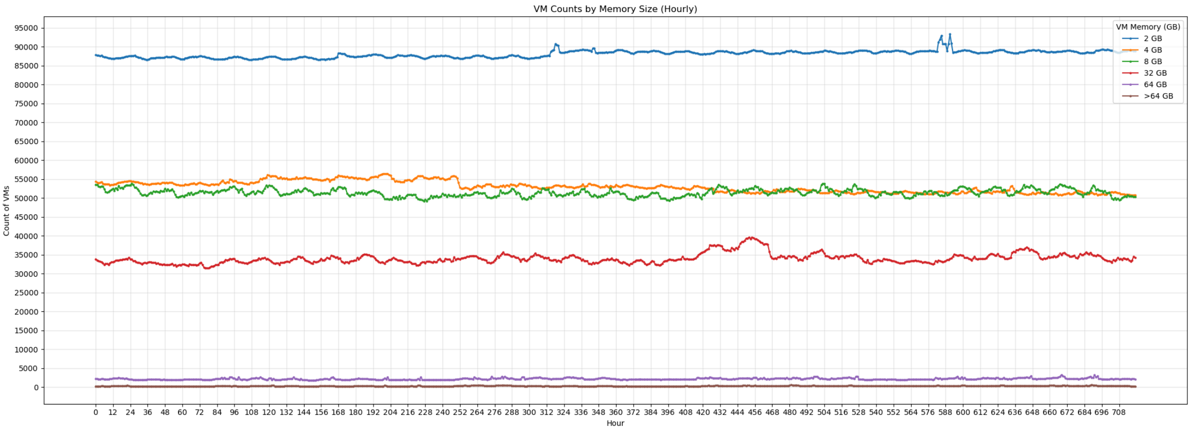
```
# Calculating hourly averages
hourly_CPU_avg = hourly_data.groupby('hour')[['max_cpu', 'avg_cpu', 'p95_ma

def plot_with_24h_means(data, col_name, color, ylabel, title):
    plt.figure(figsize=(22, 8))
    plt.plot(data['hour'], data[col_name], color=color, linewidth=1.5)
    overall_mean = data[col_name].mean()
    plt.axhline(y=overall_mean, color='green', linestyle='--', linewidth=0.

    max_hour = data['hour'].max()
    for start in range(0, max_hour + 1, 24):
        end = start + 24
        block = data[(data['hour'] >= start) & (data['hour'] < end)]
        if not block.empty:
            block_mean = block[col_name].mean()
            plt.hlines(y=block_mean, xmin=start, xmax=min(end - 1, max_hour

    plt.title(title)
    plt.xlabel('Hour (from timestamp 0)')
    plt.ylabel(ylabel)
    plt.xticks(ticks=np.arange(0, max_hour + 1, 12))
    plt.grid(True, alpha=0.3)
    plt.legend()
    plt.tight_layout()
    plt.savefig(f'azure_plots/Azure_{col_name}_plot')
    plt.show()

# Plotting CPU Utilization
plot_with_24h_means(data=hourly_CPU_avg, col_name='max_cpu', color='dodgerb
plot_with_24h_means(data=hourly_CPU_avg, col_name='avg_cpu', color='orange',
plot_with_24h_means(data=hourly_CPU_avg, col_name='p95_max_cpu', color='med
```



In [16]:
```
vm_counts = (hourly_data.groupby(['hour'])['vm_id'].nunique().reset_index(na
vm_counts = vm_counts.dropna(subset=['hour', 'num_vms'])
plot_with_24h_means(data=vm_counts, col_name='num_vms', color='brown', ylabe
```

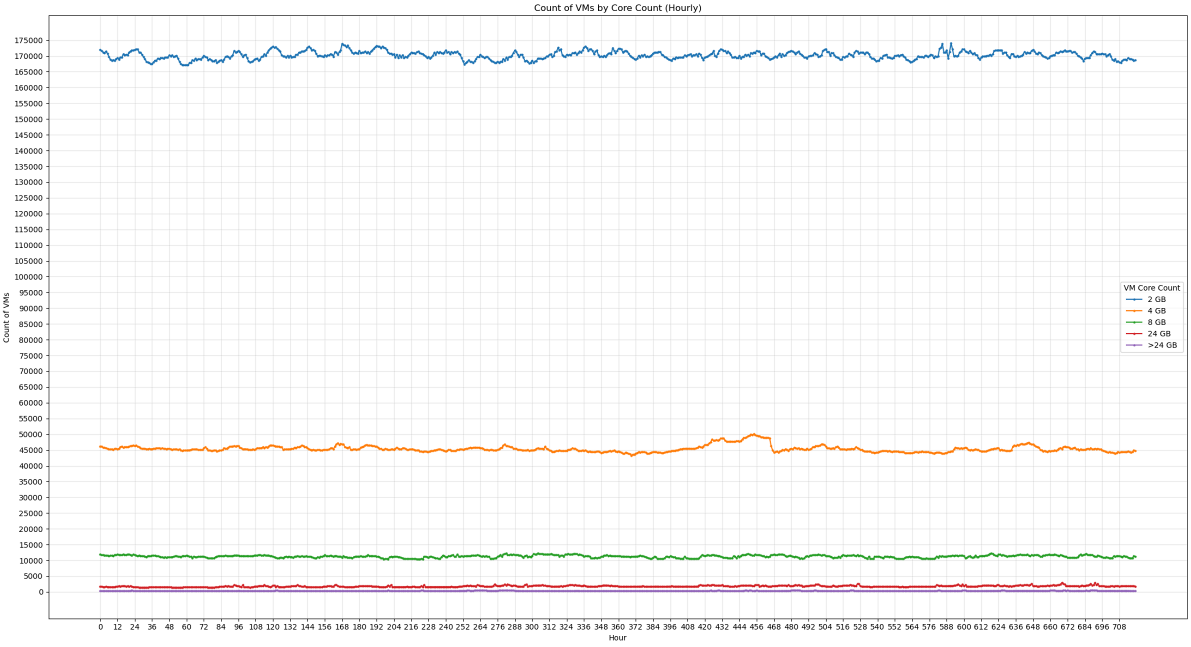


In [17]:
```
vm_order = ['2', '4', '8', '32', '64', '>64']
hourly_data['vm_memory'] = pd.Categorical(hourly_data['vm_memory'], categor
# Grouping by hour and vm_memory size to get the count of VMs
memory_counts = hourly_data.groupby(['hour', 'vm_memory']).size().unstack(f
plt.figure(figsize=(22, 8))
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']
for vm, color in zip(vm_order, colors):
    plt.plot(memory_counts['hour'], memory_counts[vm], label=f'{vm} GB', co

plt.title('VM Counts by Memory Size (Hourly)')
plt.xlabel('Hour')
plt.ylabel('Count of VMs')
plt.xticks(ticks=np.arange(0, memory_counts['hour'].max() + 1, 12))
plt.yticks(np.arange(0, memory_counts[vm_order].values.max() + 5000, 5000))
plt.grid(True, alpha=0.3)
plt.legend(title='VM Memory (GB)')
plt.tight_layout()
plt.savefig('azure_plots/Azure_vm_memory_counts_per_hour.png')
plt.show()
```



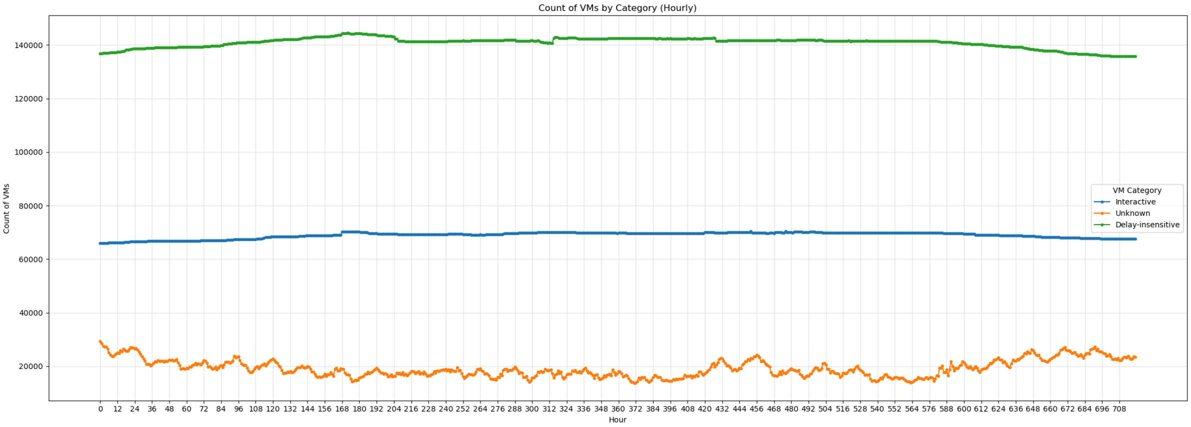
In [18]:

```
vm_core_order = ['2', '4', '8', '24', '>24']
hourly_data['vm_core_count'] = pd.Categorical(hourly_data['vm_core_count'],
# Grouping by hour and vm_core_count size to get the count of VMs
core_counts = hourly_data.groupby(['hour', 'vm_core_count']).size().unstack

plt.figure(figsize=(22, 12))
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']
for vm, color in zip(vm_core_order, colors):
    plt.plot(memory_counts['hour'], core_counts[vm], label=f'{vm} GB', colo

plt.title('Count of VMs by Core Count (Hourly)')
plt.xlabel('Hour')
plt.ylabel('Count of VMs')
plt.xticks(ticks=np.arange(0, core_counts['hour'].max() + 1, 12))
plt.yticks(np.arange(0, core_counts[vm_core_order].values.max() + 5000, 500
plt.grid(True, alpha=0.3)
plt.legend(title='VM Core Count')
plt.tight_layout()
plt.savefig('azure_plots/Azure_vm_core_counts_per_hour.png')
plt.show()
```



In [19]:

```
vm_categories = ['Interactive', 'Unknown', 'Delay-insensitive']
hourly_data['vm_category'] = pd.Categorical(hourly_data['vm_category'], cate
# Grouping by hour and vm_category to get the count of VMs
category_counts = hourly_data.groupby(['hour', 'vm_category']).size().unstac
plt.figure(figsize=(22, 8))
colors = ['#1f77b4', '#ff7f0e', '#2ca02c']

for cat, color in zip(vm_categories, colors):
    plt.plot(category_counts['hour'], category_counts[cat], label=cat, colo

plt.title('Count of VMs by Category (Hourly)')
plt.xlabel('Hour')
plt.ylabel('Count of VMs')
plt.xticks(np.arange(0, category_counts['hour'].max() + 1, 12))
plt.grid(True, alpha=0.3)
plt.legend(title='VM Category')
plt.tight_layout()
plt.savefig('azure_plots/Azure_vm_category_counts_per_hour.png')
plt.show()
```



Predictive Analysis -

In [20]:

```
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import root_mean_squared_error
```

In [45]:

```
vm_table_data = pd.read_csv("vm_table_data_azure.csv")
vm_table_data.head()
```

Out[45]:

	Unnamed: 0	vm_id	
0	0	rKggHO/04j31UFy65mDTwtjdMQL/G03xWfl3xGeiilB4/W...	ub4ty8ygwOECrIz7eaZ
1	1	YrR8gPtBmfNaOdnNEW5If1SdTqQgGQHEnLHGPjySt53bKW...	9LrdYRcUfGbmL2fFf
2	2	xzQ++JF1UAkh70CDhmzkiOo+DQn+E2TLErCFKEmSswv1pl...	0XnZZ8sMN5HY+Yg+0c
3	3	vZEivnhabRmImDr+JqKqZnpIM3WxtypwoxjfjnklR/idyR...	HUGaZ+piPP4eHjycC
4	4	MqvcZ6Au5oul6if56MJHmoSqHtX8oRv0dPkaxCld3aUcr1...	p14cXGYqCKCcF7b7Od

In [46]:

```
vm_table_data["vm_creation_hour_of_day"] = (vm_table_data["vm_creation_times
vm_table_data["total_time_running"] = vm_table_data["vm_deletion_timestamp"]
vm_table_data.head()
```

Out[46]:

	Unnamed: 0	vm_id	
0	0	rKggHO/04j31UFy65mDTwtjdMQL/G03xWfl3xGeiilB4/W...	ub4ty8ygwOECrIz7eaZ
1	1	YrR8gPtBmfNaOdnNEW5If1SdTqQgGQHEnLHGPjySt53bKW...	9LrdYRcUfGbmL2fFf
2	2	xzQ++JF1UAkh70CDhmzkiOo+DQn+E2TLErCFKEmSswv1pl...	0XnZZ8sMN5HY+Yg+0c
3	3	vZEivnhabRmImDr+JqKqZnpIM3WxtypwoxjfjnklR/idyR...	HUGaZ+piPP4eHjycC
4	4	MqvcZ6Au5oul6if56MJHmoSqHtX8oRv0dPkaxCld3aUcr1...	p14cXGYqCKCcF7b7Od

In [47]:

```
def clean_metrics_with_initial_signs(metric_value):
    if metric_value[0] == ">" or metric_value[0] == "<":
        return metric_value[1 : ]
    else:
        return metric_value

vm_table_data["vm_core_count"] = vm_table_data["vm_core_count"].apply(clean_
vm_table_data["vm_core_count"] = vm_table_data["vm_core_count"].astype("floa
```

In [48]:

```
vm_table_data["vm_memory"] = vm_table_data["vm_memory"].apply(clean_metrics_
vm_table_data["vm_memory"] = vm_table_data["vm_memory"].astype("float")
vm_table_data["vm_category"] = vm_table_data["vm_category"].astype("category
```

In [ ]:

```
training_data_X, testing_data_X, training_data_Y, testing_data_Y = train_tes
```

In [49]:

```
xgboost_regressor_model = XGBRegressor(n_estimators = 1500, enable_categori
```

In [50]:

```
xgboost_regressor_model.fit(training_data_X, training_data_Y)
avg_cpu_prediction_values = xgboost_regressor_model.predict(testing_data_X)
```

In [51]:

```
root_mean_squared_error(testing_data_Y, avg_cpu_prediction_values)
```

Out[51]:

```
14.87183564508844
```

In [52]:

```
diff_in_prediction_vals_from_truth = (abs(avg_cpu_prediction_values - testin
prediction_in_range_counter = 0
for curr_diff in diff_in_prediction_vals_from_truth:
    if curr_diff <= 10:
        prediction_in_range_counter = prediction_in_range_counter + 1
```

```
model_avg_cpu_pred_accuracy = prediction_in_range_counter * 100 / len(diff_
print("Model's Average CPU Utilization Precition accuracy is:", str(model_a
```

```
Model's Average CPU Utilization Precition accuracy is: 65.30442653845583%
```