
tubi

LARGE SCALE DISTRIBUTED VIDEO PROCESSING WITH OTP

Haofei Wang
Director of Engineering @ Tubi

What is Tubi?

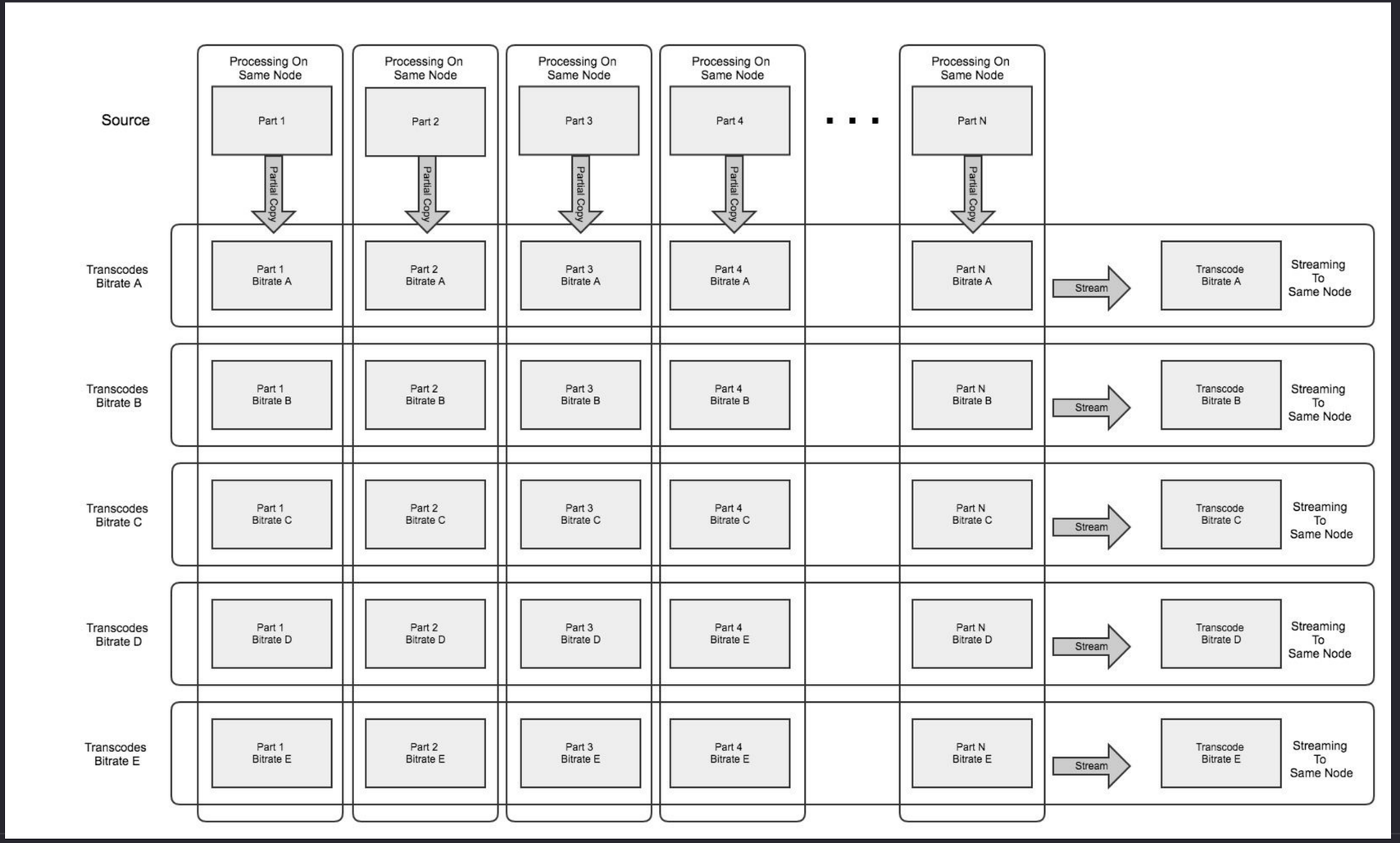
- VOD streaming service
- 100% free
- Over 25 different devices
- In-stream advertising
- Massive content library, over 20k movies and tv shows

Video Processing

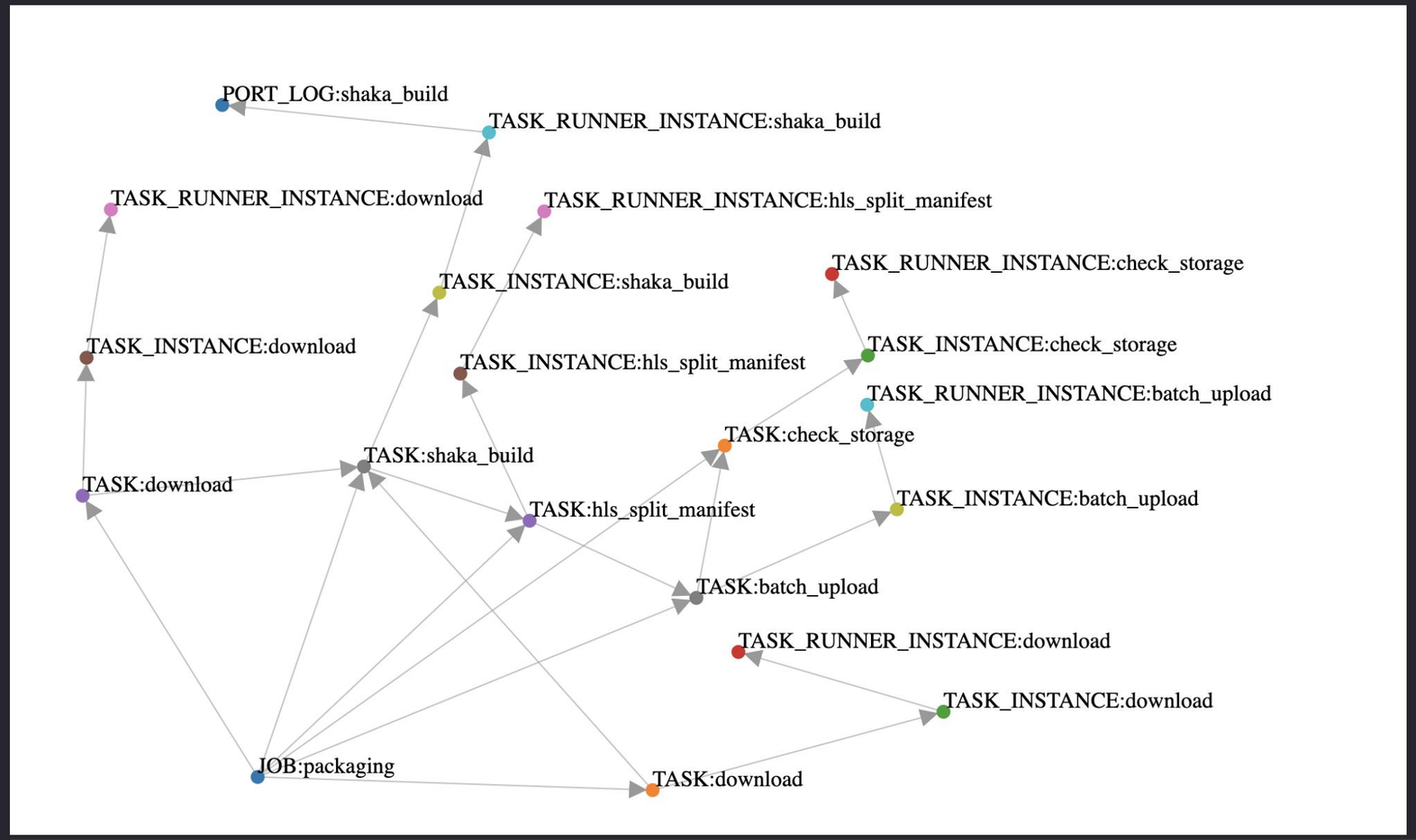
- High quality sources. Minimally compressed source videos, ranging from 50GB to 600GB.
- Massive amount of content ingestions.
- Need to support over 25 devices.

Parallel Processing

- Process each small chunk in parallel.
- One video can be broken down to thousands of tasks to process in parallel
- CPU heavy tasks
- IO heavy task
- Network heavy tasks
- Tasks have dependencies.



Parallel Processing



Why Elixir/Erlang OTP?

Supervisor: monitor the health of the child process, restart as needed

```elixir
@spec start_link(node(), map()) :: GenServer.on_start()
def start_link(node_name, args) do
  {:ok, pid} = :rpc.call(node_name, GenServer, :start, [__MODULE__, args])
  Process.link(pid)
  {:ok, pid}
end
```

Why Elixir/Erlang OTP?

Supervisor: monitor the health of the child process, restart as needed

```
def handle_info({:EXIT, pid, :normal}, state) do
  if Models.Task.unfinished_count(state.job_id) == 0 do
    {:stop, :normal, state}
  else
    {:noreply, %{state | tasks: Map.drop(state.tasks, [pid])}}
  end
end
```

Why Elixir/Erlang OTP?

Supervisor: monitor the health of the child process, restart as needed

```
def handle_info({:EXIT, pid, _error}, state) do
  {task_id, rest_tasks} = Map.pop(state.tasks, pid)
  with true <- not is_nil(task_id),
       task = Models.Task.get_by_id(task_id),
       true <- task.rest_restarts > 0 do
    {:noreply, %{state | tasks: rest_tasks}}
  else
    false ->
      {:stop, {:shutdown, :job_terminated}, %{state | tasks: rest_tasks}}
  end
end
```

Why Elixir/Erlang OTP?

Cluster is built-in: adding new node is simple, remote process is seemingless

```
@impl GenServer
def handle_info(:register, %{manager_node_name: mgr_node, quota: quota} = state) do
  mgr_node
  |> :rpc.call(Manager.Register, :register, [self(), Node.self(), quota])
  |> case do
    {:ok, pid} ->
      _ = Logger.info("Connected to Manager node #{mgr_node}")
      {:noreply, Map.put(state, :manager_pid, pid)}
    _ ->
      Process.send_after(self(), :register, 2000)
      {:noreply, state}
  end
end
```

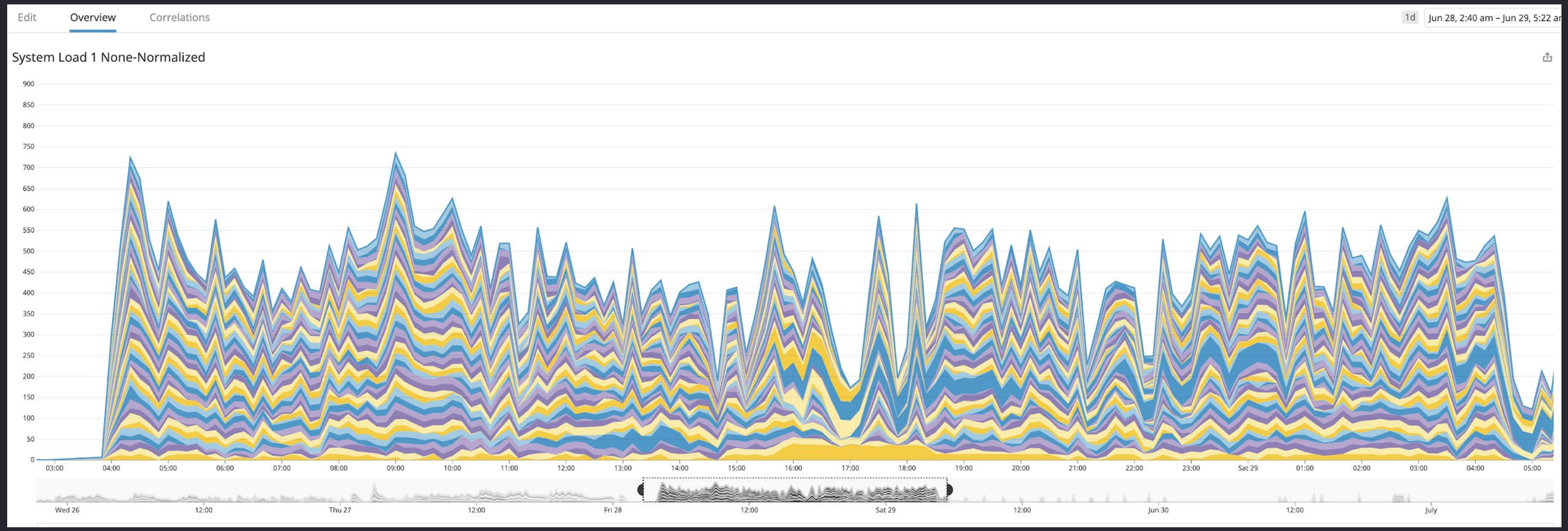
Why Elixir/Erlang OTP?

Cluster is built-in: adding new node is simple, remote process is seamless

```
@spec start_link(node(), map()) :: GenServer.on_start()
def start_link(node_name, args) do
  {:ok, pid} = :rpc.call(node_name, GenServer, :start, [__MODULE__, args])
  Process.link(pid)
  {:ok, pid}
end
```

When a batch of content need to be processed

Processed 900+ videos in 28 hours. 700+ CPU cores running at full speed. It is 1.87 minutes for each video processing



Q&A

Thank You.