]pexip[

Why keep a thread ~~running~~ idling?

Meet `GstbaseIdleSrc`

Camilo Celis Guzman
camilo@pexip.com

GStreamer Conference – October 2025

Story time!

- **Pexip's Core Characteristics:**
  - Transcoding-centric architecture.
  - Interop (To/From anything and everything™)
  - Large enterprice customers → Large-scale deployments.
  - Distributed system across multiple regions.
  - Many concurrent meetings, often with **tens** of participants per call.

- **Issue**: "Signal 6, on conference main entry point"
  - 🔥 CPU usage skyrocketed
  - 💾 Memory pressure → OOM killer almost triggered
  - 💤 System idle time = 0%
  - Result: Sadness 😭

The Problem

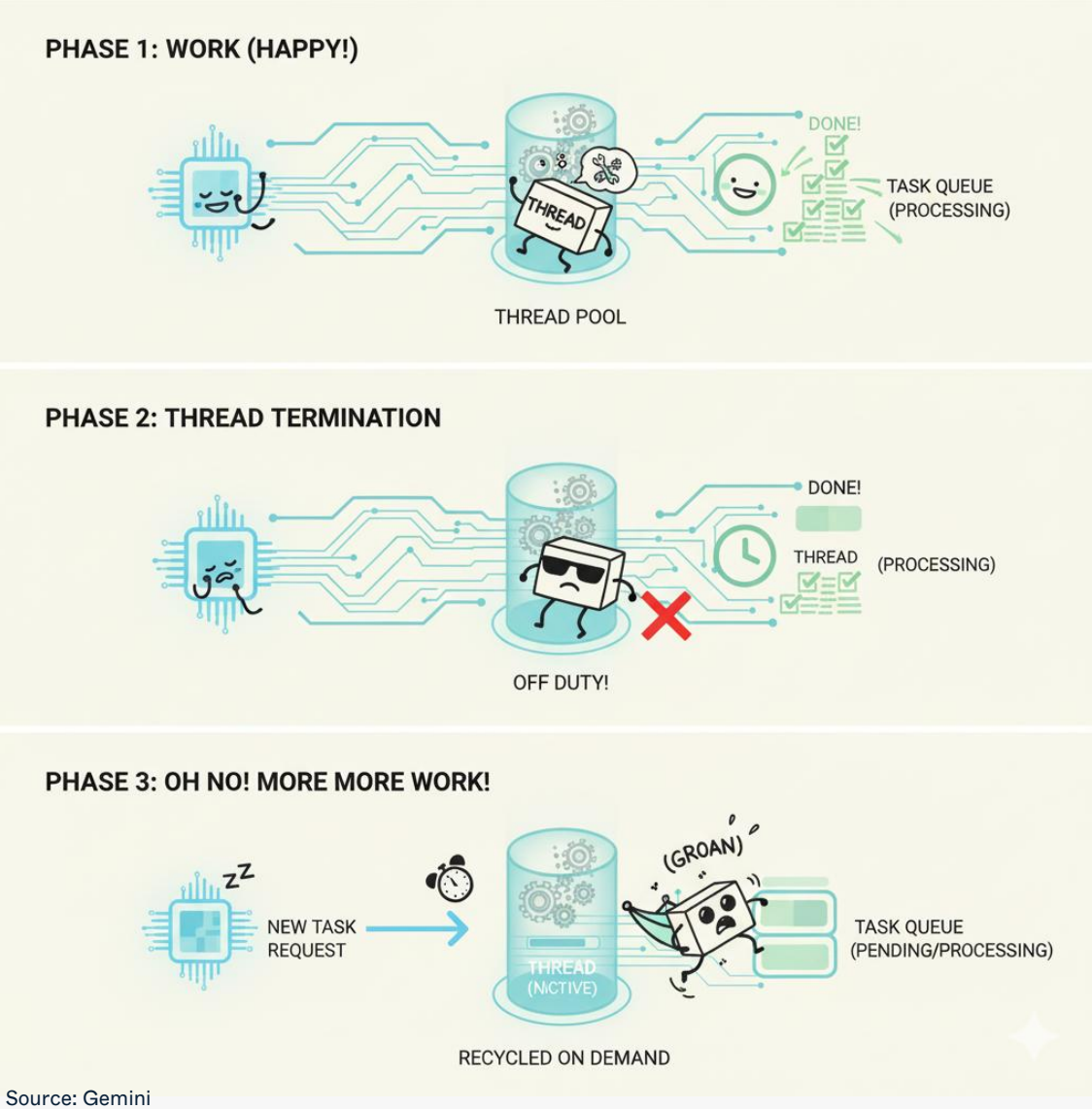
- **Graphics source element!**
  - Built from a base source class.
  - Logic: constructs some graphics, pushes once (externally controlled).
  - Single thread that becomes idle (in the OS) until new data is provided
  - **Rarely** new data is generated (max. couple of times in the lifetime of a meeting).

    - CPU overhead:
      - virtually none **if** properly programmed (I/O, timer, sync primitives).
      - Management overhead
    - Memory overhead: fixed
      - Stack
      - Control block
      - Local storage
    - **Thread limit OS/Kernel**

      ⚠️ Thread re-creation are often far more costly than context switches.

The Solution

- New source class: **GstBaseIdleSrc!**
  - Similar to GstBaseSrc / GstPushSrc
  - Push only mode
  - Non-live source
  - Internal short-lived thread that handles streaming of data



Source: Gemini

From {*Base,Push*}*Src* → *BaseIdleSrc*

```
def thread_func (...) {
  while (not flushing and not EOS) {
    // push mode
    subclass.create (buf) // ← caller's problem
    push (buf)
  }
}
```

```
def submit (...) {
  queue.append (buf)
  start_tasK (process_queue) // ← thread per buffer(s)
}

def process_queue (...) {
  for (buf : queue) {
    push (buf)
  }
}
```

Other options?

	GstBaseSrc	GstPushSrc	GstBaseIdleSrc
Purpose	Generic abstract class	Constrained source	Idle or on-demand sources
Data Flow	Push / Pull	Push only	Push-like (idle-driven)
Thread Model	No internal thread by default	Internal thread create() → push()	Short-lived on-demand thread(s)
Subclass Responsibility	fill(), create()	create()	queue_object()
Use cases	File sources, random-access	Live on continious data sources	Opportunistic / low-activity sources
Thread overhead	Low; depends on subclass	Low	Potentially high if pushes are frequent

Next steps

- Thread pools
- Consider `threadshare` concepts (async runtime)!
- Smarter scheduling / pacing of buffers
- Consider other elements that could benefit from it (GstAppSrc maybe?)
- Extended performance test – comparisons of different scenarios

🔗 **github.com/pexip/gstreamer**

Thank you!

Q & A

]pexip[

github.com/pexip/gstreamer