In this paper, the authors propose a new set of abstractions to support GPUs called **PTask API** which supports dataflow programming. The introduction of these abstraction with a gestural interface can enable us to perform enhanced system performances and data movements better than current GPU models. GPUs have surpassed CPUs as a source of high-density computing device due to the emergence of general-purpose GPUs (GPGPU) framework like CUDA. However, In the current system GPU use in application is **limited** to certain domains due to its difficulty and modularity and it’s time to see GPUs as computational devices like CPUs. GPUs have severe limitations and thus proposed several kernel-level interfaces for managing interactive, high-compute devices.

**PTask API** allows programmer write code and manage a graph-structured which have I/O ports exposing data sources and are connected by channels, which represent a data flow edge in the graph. **Kernel-level PTasks** have data movement optimization which is currently not present in current GPU. The data flow programming model supported by the PTask API delivers throughput improvements up to 4*×* across a range of microbenchmarks and a 5*×* improvement for our prototype gestural interface. The **motivation** for the kernel abstraction was the interactive gesture recognition. For data movement Xform, filters were used for picking up speeds and reading data from kernel-space into user space and buffer back into kernel-space and transfer to GPU and back. This paper also shows the overhead caused per image frame by runtime system thus spending more time migrating data. The **scheduling problem** is GPU cause the system to pause and CPU works interferes with GPU throughout.

In this paper, PTask was **designed** for three main goals:1. To bring isolation and fairness by bringing GPUs under single resource manager. 2. Programming the model focusing on application level like algorithms and data flow rather than devices, I/O and memory spaces. 3. A programming environment that allows code to be fast and modular.

**PTask API** are built on OS-level abstraction which consist of new system calls like PTask, Port, Channel, Graphs, Data blocks and Template. The **dataflows through a PTask graph** as discrete data blocks, moving through the channels, and arriving at the ports representing PTask. The **PTask invocation** can be in one of the four states namely: Waiting, Queued, Executing or Completed. The GPU completes the execution before a downstream channel drain. Further, the **Gestural interface PTask** graph is explained with a dataflow graph with multiple advantages such as avoids unnecessary communication and supports concurrent programming.