Paper summary:

Small coflows usually deliver messages for cloud data-intensive applications that are latency-sensitive. Therefore, they need to be scheduled and processed promptly. The state-of-the-art schedulers are not specifically adapted for small coflows because either they are centralized, which brings a not negligible overhead; they require prior flow information that is not always available beforehand; or they require some functionalities that are not supported by commercial off-the-shelf (COTS) switches. This paper proposes OPTAX, a decentralized and adaptive scheduling for small coflows that meet the limits of the state-of-the-art schedulers. OPTAX leverages the sendbuffer information in the host's kernel, identifies small coflows, and assigns high priorities to their flows.

Strengths:

- The paper deals with tail-latency workflows which are critical especially nowadays with the FaaS context
- Experiment results suggest quite some improvement

Weaknesses:

- The work is not well motivated and the positioning claim in the introduction is not substantiated: the reference used by the authors to support that small coflow is dominant in the cloud states exactly the opposite.
- The challenges presented are specific to coflows in general, not to small coflows which is the focus of the paper
- Only one application use-case
- Must important, the paper seems a very tiny modification of OPTAS (Infocom 2016).

Comments to the authors:

++ *Motivation*

1. The transition between the example depicted in figures 1a and 1b to *Therefore, it is well accepted that all small […]* is not straightforward since all flows in that example have the same execution time, none of which is said to be small or large.
2. It is unclear how the way the challenge is expressed. The example of Varys is not pretty clear: why does it ignore coflows of sizes lower than 25MB?
3. The paper [16] on Varys that you reference to support the fact that small coflow is dominant in the cloud claims exactly the opposite: *[…] small coflows contribute less*

than 1% of the traffic in data-intensive clusters […]

4. Apart from the first challenge, it is unclear how the two other ones are specifically related to **small** coflows which is the focus of the paper. My understanding is that the authors wanted to point out in which way existing schedulers are not adapted for small coflows. If so, it would be better to explicitly specify that these limits are related to the state-of-the-art, not to small coflows scheduling in general.

++ *Analysis*

1. I kindly suggest that the requirements presented in this section be moved to the motivation section, precisely, to drive the challenges related to the state-of-the-art.

++ *Design*

1. Tasks with the same ID are considered concurrent, but how are they identified from each other since I assume they have the same src and dest, and the flow-size is said not to be always known in advance?
2. I suggest that the authors clearly state, from the introduction, how they define a small coflow. Is it a fixed generally well-known size (as 1MB in the case of the paper) or a defined value depending on the implementation?
3. jprobe and netfilter hooks are said to be used for performing both tasks and buffer monitoring, but in Fig.3 these components are only visible on the receiver side and a connection is made only with the task monitor. This makes it difficult to follow the design description.
4. Can the authors not only give a summary definition of what jprobe and netfilter are but better explain how they are integrated and leveraged in the task and buffer monitors communication?
5. The description of the buffer monitor is somewhat disordered and not well structured and thus, does not allow for a good understanding of the scheduling flow.

++ *Collaborative Scheduling*

1. [5.2]: what are “*show*” flows?
2. Fig. 6,7,8: should “tiny tasks” in the figures be read as “small coflows”?
3. Fig. 7 & 8 can be placed side by side on the same line
4. What is the meaning of DAG in DAG-aware schedulers? This is never defined and first referenced in 5.1.

++ *Evaluation*

1. Fig. 13 & 14 are not uniform to other figures. It would be better if they are plotted using the same tools as for others graphs.

Infocom'16 paper extension

- I kindly suggest that OPTAX should be explicitly presented as an extension on OPTAS, and not a new tool since the challenges and design of both are the same
- Section 5: the authors present the integration of OPTAX with the existing framework as an improvement while this was already the case with OPTAS. Indeed, the results presented in this section (Figures 13 & 14) are the same as Section-V.B of the Infocom'16 paper (Figures 9 & 10)
- Section 6: presents no graph results
- In general, the improvement with the original work is not quite substantial and the ideas do not seem to be better explained.

Typos:

- Baraat spelling: sometimes *Baraat* sometimes *Barratt*
- Page 3, Introduction - word redundancy: *Experimental evaluation shows that **shows that** […] under **under** […]*
- Page 4, Section-2.(1) - probably missing preposition: *To achieve centralized […] to the scheduler, then calculate […]* => is there not a missing "who/which" before "then"? Are the sender/receiver that calculates the policy or the scheduler?
- Page 8, Section-4.2: *coflow"a"*
- Page 9, Section-4.4: *Calcul-ator* (is this a copy+paste?)
- Page 10, Section-4.4.2: a dot is missing at the end of the sentence *[…] sometimes the send buffer […] in advance*
- Page 15, Section-7.2.1: internal instead of interval
- Page 15, Section-7.2.1: what does *expedietic* mean in *expedietic rate*