3. Анісімов А.В., Кулябко П.П., Терещенко В.М. Паралельні алгоритми в дослідженнях неперервних систем. – К.: РВЦ Київський університет, 1999. – 55 с.

4. Антонов А.С., Воеводин Вл.В. Новый подход к построению методов межпроцедурного анализа программ // Материалы международной конференции УкрПРОГ-98. – К. – 1998. – С. 77-84.

5. Мельник А.О., Яковлєва І.Д. Подання потокового графа алгоритму структурною матрицею // Вісник Хмельницького національного університету. – 2008. – № 4. – С.124-129.

6. Рабкин Е.Л., Фарфоровская Ю.Б. Кафедра высшей математики СПбГУТ им. проф. М.А. Бонч-Бруевича Дискретная математика. Булевы функции и элементы теории графов // http://dvo.sut.ru/libr/himath/w163rabk/

7. Ахо А., Сети Р., Ульман Дж. Д. Компиляторы: принципы, технологии и инструменты. М.: Вильямс, 2001. – 768 с.

Надійшла 8.12.2009 р.

УДК 004.451

С.В. МОСТОВИЙ, О.Л. КОВТУН, І.В. ПРОКОПИШЕН
Хмельницький національний університет

МЕТОД ТА ЗАСОБИ ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ В ПЕРСОНАЛЬНОМУ КОМП'ЮТЕРІ

Розроблено метод прогнозування стану процесів в персональних комп'ютерах, який дає можливість передбачати настання ситуації взаємоблокування процесів на основі аналізу взаємодії сигнатур процесів.

The method of forecasting of a status of processes in personal computers is developed. It enables to provide approach of deadlock of processes based on analysis of interaction of their signatures.

Ключові слова: взаємоблокування процесів, прогнозування стану процесів.

Вступ

При паралельному виконанні задач можуть виникати такі ситуації, при яких два і більше процеси весь час знаходяться в стані блокування, очікуючи ресурсів системи, які в даний момент утримує інший процес, що, в свою чергу, теж знаходиться в стані блокування. Про такі процеси говорять, що вони знаходяться в стані взаємного блокування, "дедлока" (deadlock) або "клінчу" (clinch). В даний час розглядаються можливі компромісні рішення з погляду накладних витрат на включення засобів боротьби з взаємоблокуваннями і очікуваних від цього вигод. В деяких випадках ціна, яку доводиться платити за те, щоб зробити систему вільною від взаємоблокувань, дуже висока. В системах реального часу виникнення ситуації взаємоблокування може призвести до катастрофічних наслідків.

Постановка задачі

Для вирішення проблеми взаємоблокування відомо багато методів і алгоритмів [1-4]. Вони поділяються на наступні групи:

- Запобігання взаємоблокуванню, що полягає у забезпеченні невиконання однієї з чотирьох умов для взаємоблокування.
- Уникнення взаємоблокування. Полягає у тому, що б не задовільняти запит на ресурс, якщо його виділення може потенційно спричинити взаємоблокування.
- Виявлення взаємоблокувань. Полягає у тому, що б завжди задовільняти запити на ресурси, коли це можливо, і періодично перевіряти систему на наявність взаємоблокувань. Якщо взаємоблокування має місце, то вирішити проблему.
- Ігнорування взаємоблокувань. Має сенс, коли ймовірність виникнення взаємоблокування дуже низька.

Вагома частка взаємоблокувань припадає на взаємоблокування процесів, що виконуються в операційних системах (ОС) для персональних комп'ютерів (ПК). Проте, методи розв’язання задачі взаємоблокування процесів в ОС ПК мають ряд недоліків [4] (блокування роботи ОС, наявність циклів активного очікування, складність програмної реалізації для багатьох процесів, необхідність використання спеціалізованої команди процесора) і є складними для їх реалізації. Тому більшість сучасних ОС не містять засобів для вирішення проблеми взаємоблокування процесів.

Для усунення виявлених недоліків та розв’язання задачі взаємоблокування процесів в [5] запропонована модель прогнозування стану процесів в персональних комп'ютерах, яка базується на використанні сигнатур процесів.

До складу моделі входять наступні структурні частини (рис. 1):

- підсистема виявлення зміни стану процесу – призначена для контролю за поточними параметрами процесів, що вже присутні на ПК в певній ОС, та запуском нових процесів;
- підсистема визначення характеристик ОС даного ПК – призначена для контролю за зміною основних характеристик, що суттєво впливають на наближення процесів до стану взаємоблокування;
- підсистема побудови сигнатури процесу – призначена для побудови сигнатури процесу, що виконує перехід у стан готовності і ще не має сигнатури, та модифікації сигнатури процесів, що змінили

значення своїх параметрів та мають відповідні їм сигнатури;

- множина сигнатур процесів, що присутні на ПК в певній ОС – призначена для зберігання сигнатур процесів, які в даний момент виконуються на ПК;
- підсистема аналізу та логічного висновку – за допомогою множини правил взаємодії сигнатур процесів виділяє із множини сигнатур всіх процесів підмножину сигнатур процесів, що наближаються до стану взаємоблокування, і робить висновок про можливість продовження роботи цих процесів.

Для здійснення аналізу та логічного висновку використаємо нечітку експертну систему. Згідно з [5,6] включимо до складу підсистеми аналізу та логічного висновку наступні компоненти:

- підсистема фаззифікації вхідних параметрів – визначає ступені впевненості в тому, що вихідні лінгвістичні змінні отримають конкретні значення;
- підсистема висновку – на основі множини правил взаємодії сигнатур процесів та набору вхідних лінгвістичних змінних проводиться оцінка істинності для кожного правила та формується єдина нечітка множина;
- підсистема дефаззифікації вихідного параметру – перетворює нечіткий набір значень вихідної лінгвістичної змінної до точного значення.

Рис. 1. Модель процесу прогнозування стану процесів в персональних комп'ютерах

Прогнозування стану процесів в ПК

Розроблений метод прогнозування стану процесів в ПК включає наступні етапи:

1) виявлення змін в системі;

2) побудова сигнатури процесів;

3) визначення основних характеристик ПК та ОС;

4) приведення до нормованого вигляду вхідних параметрів та їх передача на підсистему аналізу та логічного висновку;

5) здійснення висновку про настання взаємоблокування процесів

На першому етапі виявляємо зміни, що відбуваються в системі, а саме виявляємо запити на створення нового процесу чи зміну параметрів вже існуючого процесу. Ця робота покладена на підсистему виявлення зміни стану процесу.

На другому етапі здійснюємо побудову (перебудова для вже існуючих процесів) сигнатури для виявленого процесу. Побудову сигнатури здійснимо наступним чином:

1) Перевіряємо, чи має процес раніше сформовану сигнатуру. Якщо так, то переходимо до дії 5

2) Отримуємо інформацію про процес, яка необхідна для формування його сигнатури.

3) Формуємо сигнатуру процесу. На цьому кроці відбувається перетворення значень необхідних параметрів процесу до заданого вигляду і утворення його сигнатури (рис.2). До необхідних параметрів входять наступні: ідентифікатор процесу (x_1 біт); ідентифікатор батьківського процесу (x_2-x_1 біт); ідентифікатор користувача (x_3-x_2 біт); пріоритет процесу (x_4-x_3 біт); кількість дескрипторів файлів, що використовуються процесом (x_5-x_4 біт); обсяг віртуальної пам'яті, що використовує процес (x_6-x_5 біт); час виконання процесу (x_7-x_6 біт); додаткові параметри процесу (x_n-x_7 біт).

4) Створену сигнатуру додаємо до множини сигнатур процесів.

5) Перевіряємо, чи змінились значення складових сигнатури процесу. Якщо так, то переходимо до дії 6, інакше переходимо до дії 7.

6) Модифікуємо сигнатуру процесу згідно з поточними значеннями складових.

7) Зберігаємо поточну множини сигнатур процесів.

Ці дії покладені на підсистему побудови сигнатури.

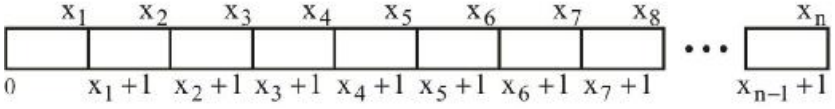


Рис. 2. Представлення сигнатури у машинному форматі

На третьому етапі визначаємо поточні характеристики ПК та ОС. Ці дії проводяться підсистемою визначення характеристик ПК та ОС. Оскільки взаємоблокування настає з причини конкуренції процесів за системні ресурси, то включаємо їх кількісні характеристики до бази знань експертної системи:

- кількість процесів користувача, присутніх на ПК в певній ОС;
- кількість процесів ядра системи, присутніх на ПК в певній ОС;
- обсяг ОП на ПК в певній ОС (загальний та вільної в даний момент);
- обсяг ЗП на ПК в певній ОС (загальний та вільної в даний момент);
- кількість пристроїв вводу-виводу інформації (загальна та вільних у даний момент);
- загальну кількість файлів на ПК в певній ОС.

На четвертому етапі передаємо множину сигнатур процесів та множину характеристик ПК та ОС для проведення аналізу на предмет настання ситуації взаємоблокування в системі. Ці дані подаються на підсистему аналізу та логічного висновку. Оскільки всі вхідні характеристики знаходяться у різних числових межах, що є незручним для опрацювання результатів, то на даному етапі проводиться нормування даних величин. Зведемо усі показники до меж [0;1]. Для цього необхідно виконати ділення показника на максимально допустиме значення цього показника в конкретній комп'ютерній системі (КС). Проте такий підхід до нормування показників не буде враховувати ступінь впливу даного показника на результат системи прогнозування стану процесів, оскільки для показників із однаковим нормованим значенням, але з різними поточними та максимально допустимими значеннями, буде різний вплив на КС. Тому необхідно при нормуванні враховувати максимально допустиме значення показника. Для цього нормування показників проведемо за наступною формулою:

$$P_n = 1 - \frac{P_d + P_m}{P_d \cdot P_m},\ P_d \neq 0, \tag{1}$$

де P_n – черговий нормований показник;

P_d – поточне значення показника в конкретній КС;

P_m – максимальне значення показника в конкретній КС.

Такий підхід до нормування показників дозволяє врахувати ступінь впливу даного показника на результат системи прогнозування стану процесів.

На п'ятому етапі відбувається аналіз взаємодії процесів на основі аналізу їхніх сигнатур та характеристик ПК та ОС і здійснюється висновок про настання ситуації взаємоблокування процесів. Для здійснення даних дій використовується нечітка експертна система [6]. На рис. 3, 4а, 4б, 4в представлено результати моделювання у системі MatLab 6.1.

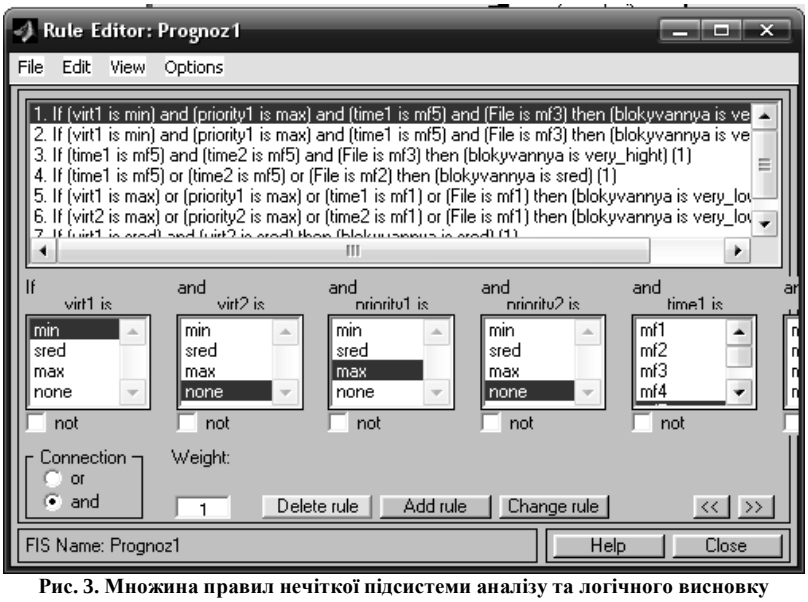
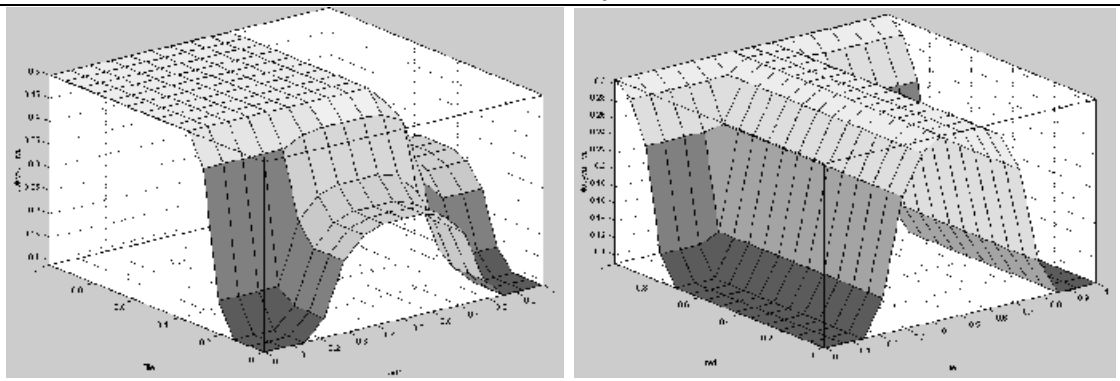
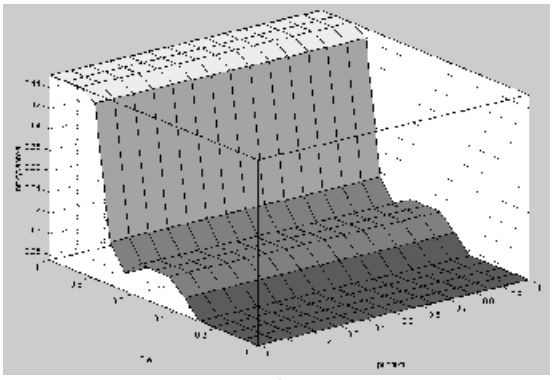


Рис. 3. Множина правил нечіткої підсистеми аналізу та логічного висновку

Позначимо через U множину всіх можливих сигнатур процесів (універсум). Виділимо множину сигнатур процесів, що є активними в даний момент в КС і позначимо її через A ($A \subset U$). Процеси, що є активними в даний момент, можна розділити на 2 підмножини: процеси, запущені користувачем, та процеси, запущені ядром системи. Отже, множину A можна розбити на дві підмножини: K та S ($K \subset U, S \subset U, K \cup S = A$). Всі перераховані множини належать до чітких. Їх характеристична функція приймає 1, якщо елемент належить множині, і 0 в протилежному випадку.



а) **б)**



в)

Рис.4. Ймовірність настання взаємоблокування

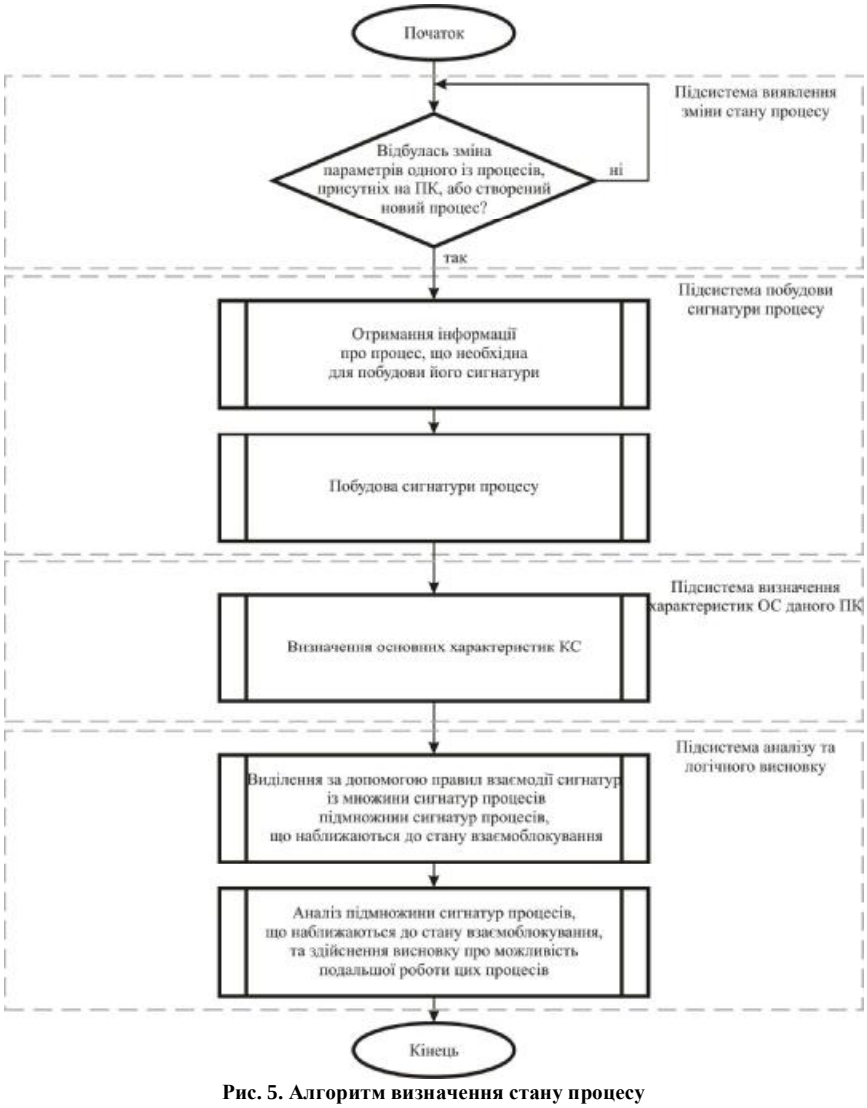


Рис. 5. Алгоритм визначення стану процесу

При наближенні стану блокування процесів в КС із множини А можна виділити підмножину А* сигнатур процесів, які найімовірніше призведуть до цього стану. Відповідно множину А* також можна розділити на 2 підмножини: К* - множина процесів, запущених користувачем, які ймовірно призведуть до стану блокування, та S* - множина процесів, запущених ядром системи, які ймовірно призведуть до блокування ($A^* \subset U, K^* \subset U, S^* \subset U, K^* \cup S^* = A^*$). Множини А*, К*, S* є нечіткими, оскільки сигнатури процесів входять у них з певними ймовірностями.

Якщо новий процес за висновком системи не призводить до взаємоблокування, то йому дозволяється виконання на ПК і його сигнатура додається до бази сигнатур працюючих процесів. В протилежному випадку система виявляє множину процесів, які наближаються до стану взаємоблокування, та приймає рішення, який із процесів зняти із виконання.

Послідовність наведених вище дій можна подати наступним алгоритмом (рис. 5):

Для оцінки ефективності розробленого методу проведено його порівняння з відомими методами уникнення ситуації взаємоблокування процесів. При порівнянні до уваги були взяті основні недоліки існуючих методів. Проведене дослідження показало, що у розробленому методі вирішується частина суттєвих недоліків відомих методів уникнення взаємоблокування. Результати дослідження подані в табл.1. Для оцінки ефективності розробленого методу було реалізовано програмне забезпечення під ОС сімейства Linux за даним методом і за алгоритмом банкіра [7]. Результати дослідження подані в табл.2

Таблиця 1

Результати порівняння розробленого методу та відомих методів

№ п/п	Параметр для порівняння	Назва методу								
		Заборона на переривання	Строге чередування	Алгоритм Деккера	Алгоритм Петерсона	Алгоритм банкіра	Семафори	Монітори	Мютекси	Прогнозування стану процесів
1	Складність реалізації для 2 і більше процесів	ні	ні	так	так	так	ні	ні	ні	ні
2	Прив'язка до кількості та видів ресурсів ПК	ні	ні	ні	ні	так	ні	ні	ні	ні
3	Активне очікування	так	так	так	так	так	так	ні	ні	так
4	Необхідність підтримки на рівні компіляторів	ні	ні	ні	ні	ні	ні	ні	так	ні
5	Наявність спеціалізованої команди процесора	ні	ні	ні	ні	ні	ні	так	ні	ні
6	Зниження продуктивності роботи обчислювальної системи	ні	так	ні	ні	ні	ні	ні	ні	ні
7	Блокування роботи операційної системи	так	ні	ні	ні	ні	так	ні	ні	ні
8	Спосіб реалізації методу	апаратний	програмний	програмний	програмний	програмний	програмний	програмний	програмно-апаратний	програмний

Таблиця 2

Результати порівняння програмного забезпечення за розробленим методом та алгоритмом "банкіра"

№ п/п	Параметр для порівняння	Алгоритм "банкіра"	Розроблені алгоритми
1	Складність реалізації для 2 і більше процесів	так	ні
2	Прив'язка до кількості та видів ресурсів ПК	так	ні
3	Завантаження центрального процесора	12-14%	15-18%
4	Кількість виявлених ситуацій взаємоблокування	5 з 10	9 з 10

Висновок

Розроблений метод прогнозування стану процесів в персональних комп'ютерах дає можливість передбачати взаємоблокування процесів на основі аналізу взаємодії сигнатур процесів та усуває основні

недоліки відомих методів, зокрема не потребує спеціалізованої команди процесора, не допускає блокування роботи ОС, є простим у реалізації для багатьох процесів під різні типи операційних систем.

Літератури

1. Coffman E.G., Elphick M,J., Shoshani A. System deadlocks // Computing Surveys, Vol.3, No.2, June 1971. – Pages: 67 – 78.

2. Nima Kaveh, Wolfgang Emmerich. Deadlock detection in distribution object systems // Software Engineering Notes, Vol.26, No.5, September 2001. – Pages: 44 – 51.

3. Saddek Bensalem, Jean-Claude Fernandez, Klaus Havelund, Laurent Mounier. Confirmation of deadlock potentials detected by runtime analysis // International Symposium on Software Testing and Analysis – Portland, Maine, USA, 2006. – Pages: 41 – 50.

4. Савенко О.С., Кльоц Ю.П., Мостовий С.В. Дослідження та аналіз блокування процесів в комп'ютерній системі // Вісник ХНУ – 2007. – №3.- Т. 1. – С.248-251

5. Савенко О.С., Мостовий С.В. Модель прогнозування стану процесів в комп'ютерній системі // Радіоелектронні і комп'ютерні системи – Харків: ХАІ, 2008. - №5 (32). – С.109-115

6. Мостовий С.В. Система прогнозування стану процесів в персональному комп'ютері // Сборник трудов VIII международной конференции "Интеллектуальный анализ информации" (ИАИ-2008) – Киев: Просвіта, 2008. – С.308-314

7. Мостовий С.В. Алгоритми і програмні засоби прогнозування стану процесів в персональному комп'ютері // Вісник ХНУ – Хмельницький: ХНУ, 2008. – №5. – С.124-130

Надійшла 17.12.2009 р.

УДК 621.515

В.В. СЛАВІН
Хмельницький національний університет

АНАЛІЗ МЕТОДІВ ВІДНОВЛЕННЯ ТУРБОКОМПРЕСОРІВ

В статті розглянуто основні причини виникнення несправностей ротора турбокомпресора та їх вплив на стан робочих поверхонь вала ротора. Проаналізовано сучасні способи відновлення робочих поверхонь вала ротора, їхні переваги та недоліки. Наведено апробований технологічний процес відновлення ротора методом ремонтних розмірів.

In the article principal reasons of origin of disrepairs of rotor of turbo-compressor and their influence are considered on the state of workings surfaces of vala rotor. The modern methods of proceeding in the workings surfaces of vala, their advantages and failings, are analysed. The technological process of proceeding in a rotor is resulted by the method of repair sizes.

Ключові слова: ротор турбокомпресора, робочі поверхні, несправності ротора, умови технічної експлуатації, спрацювання, знос, способи відновлення.

Турбокомпресор є агрегатом, який перетворює кінетичну енергію відпрацьованих газів в обертовий момент коліс ротора (40 – 250 тис. об/хв$^{-1}$), одне з яких подає стиснуте повітря до циліндрів двигуна, в результаті отримуємо більшу потужність двигуна, меншу витрату пального, а відпрацьовані гази мають меншу кількість шкідливих речовин, які надходять в атмосферу із випускного трубопроводу. В разі порушення роботи систем двигуна та недотримання правил і умов експлуатації турбокомпресорів виникають несправності. Несправності виникають внаслідок збільшення радіальних та осьових зазорів в спряжених деталях (вал ротора – втулка підшипник), якщо їх вчасно не виявити, тоді будуть мати місце граничні пошкодження деталей турбокомпресора, які зменшать ймовірність використання в подальшому переваги турбокомпресора. Граничні зазори виникають через недостатню кількість масла на робочих поверхнях та неякісну його подачу, або масло забруднене абразивними частинками через неперіодичну заміну масла чи погану якість фільтрування. Забруднене масло призводить до абразивного зносу, що є результатом ріжучої або дряпаючої дії на робочі поверхні деталі. В якості абразивних частинок виступають часточки двоокису кремнію (пісок), окисли алюмінію, які мають твердість значно більшу ніж метал пар тертя. Розмір частинок рівний з величиною зазору і коливається в межах 5…120 мкм у вигляді пилу. Забруднення дрібними частинками масла не виявляється візуально, проте викликає підвищений знос підшипників, коли абразивна часточка розташовується між двома поверхнями тертя. Через спрацювання циліндро-поршневої групи двигуна відбувається забруднення масла пальною сумішшю (хімічне забруднення), що викликає перегрів підшипників, зношення вала ротора турбокомпресора. Вся негативна дія вище несправностей внаслідок недотримання умов та правил технічної експлуатації призведе до спрацювання робочих поверхонь основних деталей та вузлів, виникнення необхідності капітального ремонту турбокомпресора. Капітальний ремонт турбокомпресорів, тому необхідно, що це агрегат, в якому не можна замінити лише одну деталь (кільце, втулку, ротор), спрацювання одної деталі говорить про спрацювання решти деталей [1].